#136 FEBRUARY 1988

2.95 (3.95 CANADA)

# Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

## Debugging on the 386

## Making Serial Links Work

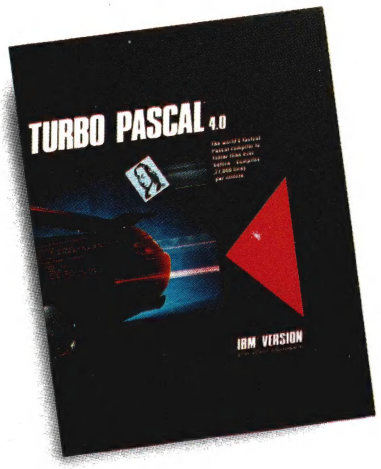
## New Product Review Section

## Languages: C, Forth, Pascal

0 38351 16562 8 02

#136 FEBRUARY 1988

# *ter* than ever before!




## *4.0 uses logical units for separate compilation*

Pascal 4.0 lets you break up the code gang into "units," or "chunks." These logical modules can be worked with swiftly and separately—so that an error in one module is seeable and fixable, and you're not sent through all your code to find one error. Compiling and linking these separate units happens in a flash because your compiling horsepower is better than 27,000 lines a minute.* And 4.0 also includes an automatic project Make.

## *4.0's cursor automatically lands on any trouble spot*

4.0's interactive error detection and location means that the cursor automatically lands where the error is. While you're compiling or running a program, you get an error message at the top of your screen *and* the cursor flags the error's location for you.

## *4.0 gives you an integrated programming environment*

4.0's integrated environment includes pull-down menus and a built-in editor. Your program output is automatically saved and shown in the output window. You can Scroll, Pan, or Page through all your output and know where everything is all the time. Given 4.0's integration, you can edit, compile, find and correct errors—all from inside the integrated development environment.

## *You'll never lose your mind, because 4.0 never loses your place*

Whenever you re-load 4.0, it remembers what you and it were doing before you left. It puts you right back in the editor with the same file and in the same place as you were working last.

*Run on an 8 MHz IBM AT.

**If within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.



| *Please check box(es)* | *Sugg. Retail* | *Upgrade Price†* | Serial No. |
|---|---|---|---|
| ☐ Turbo Pascal 4.0 Compiler | *$ 99.95* | *$ 39.95* | _____ |
| ☐ Turbo Pascal Tutor | *69.95* | *19.95* | _____ |
| ☐ Turbo Pascal Database Toolbox | *99.95* | *29.95* | _____ |
| ☐ Turbo Pascal Graphix Toolbox | *99.95* | *29.95* | _____ |
| ☐ Turbo Pascal Editor Toolbox | *99.95* | *29.95* | _____ |
| ☐ Turbo Pascal Numerical Methods Toolbox | *99.95* | *29.95* | _____ |
| ☐ Turbo Pascal Gameworks | *99.95* | *29.95* | _____ |

Total product amount $ _____
CA and MA residents add sales tax $ _____
In US please add $5 shipping and handling for each product ordered $ _____
Outside US please add $10 shipping and handling for each product ordered $ _____
Total amount enclosed $ _____

Please specify diskette size: ☐ 5¼" ☐ 3½"

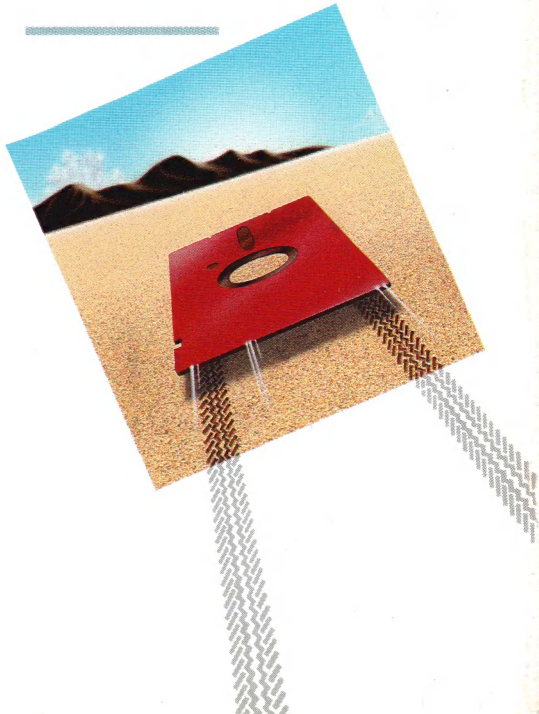
Payment: ☐ VISA ☐ MC ☐ Check ☐ Bank Draft

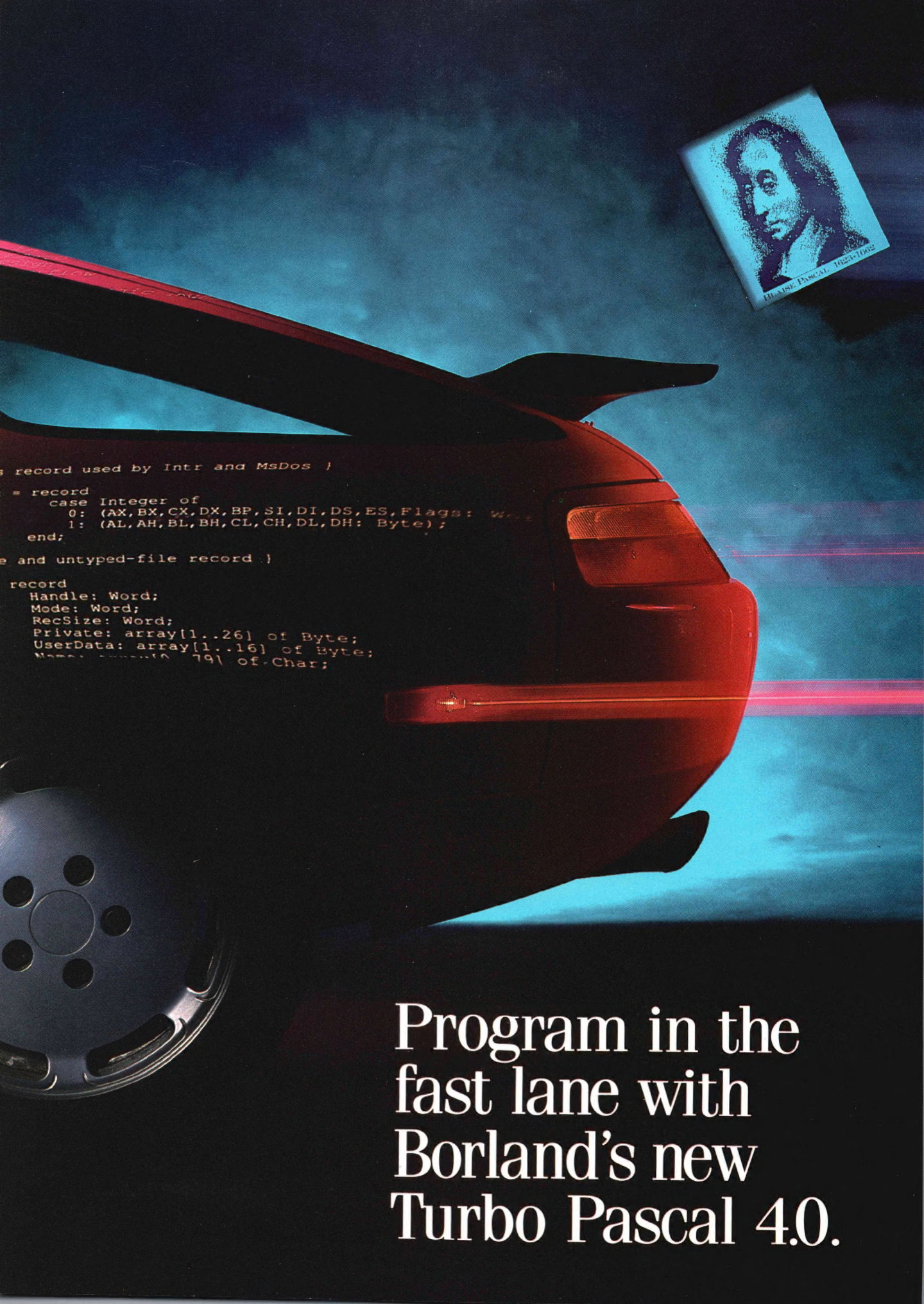
Credit card expiration date: _____/_____

Card # |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

†To qualify for the upgrade price you must give the serial number of the equivalent product you are upgrading.

**DDJ 2/88**




; record used by Intr and MsDos }
= record
case Integer of
0: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags:
1: (AL,AH,BL,BH,CL,CH,DL,DH: Byte);
end;
e and untyped-file record }
record
Handle: Word;
Mode: Word;
RecSize: Word;
Private: array[1..26] of Byte;
UserData: array[1..16] of Byte;
of Char;
BLAISE PASCAL 1623-1662
Program in the fast lane with Borland's new Turbo Pascal 4.0.

# The fast lane is *fas*

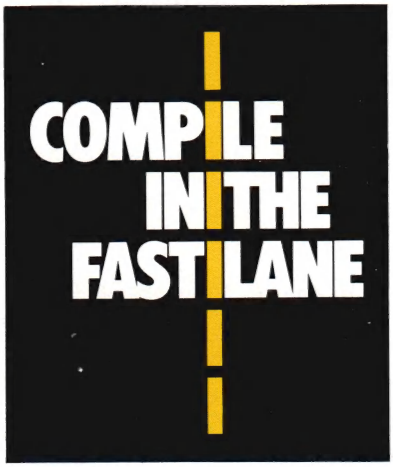
Our new Turbo Pascal® 4.0 is so fast, it's almost reckless. How fast? Better than 27,000 lines of code per minute. That's much faster than 3.0 or any other Pascal compiler and the reason why you need 4.0 today.

## *Pascal. The fastest and the best.*

If you're just now learning a computer language, learn Pascal. If you're already programming in Pascal, you're programming with a winner because Pascal is the worldwide language of choice. Pascal is the most popular language in university computer science classes and with computer enthusiasts who appreciate Pascal's modern programming structure. It's powerful, coherent, easy to learn and use—and with Turbo Pascal 4.0—faster than ever before.

## *Turbo Pascal: Technical excellence*

Commitment to technical excellence and




superiority also means commitment to detail, however painstaking, and that takes time. 4.0's predecessor, Turbo Pascal 3.0 is the worldwide standard, and with Turbo Pascal 4.0, we've bettered that standard. 4.0 is clearly the world's fastest development tool for the IBM® PS/2 series, PC's and compatibles—and the world's favorite Pascal compiler.

## *4.0 breaks the code barrier*

No more swapping code in and out to beat the 64K code barrier. Designed for large programs, Turbo Pascal 4.0 lets you use all 640K memory in your computer. You paid for all that memory, now you can use it freely.

For the IBM PS/2 and the IBM and Compaq families of personal computers and all 100% compatibles.

# YES!

**I want to upgrade to Turbo Pascal 4.0 and the 4.0 Toolboxes**

Registered owners have been notified by mail. If you are a registered Turbo Pascal user and have not been notified of Version 4.0 by mail, please call us at *(800) 543-7543*. To upgrade if you have not registered your product, just send the original registration form from your manual and payment with this completed coupon to:

*Pascal 4.0 Upgrade Dept.*
*Borland International*
*4585 Scotts Valley Drive*
*Scotts Valley, CA 95066*

Name _____

Ship Address _____

City _____ State _____

Zip _____ Telephone ( ) _____

This offer is limited to one upgrade per valid registered product. It is good until June 30, 1988. Not good with any other offer from Borland. Outside U.S. make payments by bank draft payable in U.S. dollars drawn on a U.S. bank. CODs and purchase orders will not be accepted by Borland.

DDJ 2/88

# Now's the time for a *fast* decision: Upgrade now to 4.0!

## *Compatibility with Turbo Pascal 3.0*

We've created 4.0 to be highly compatible with version 3.0 and included a conversion program and compatibility units to help you convert all your 3.0 programs to 4.0.

## *Highlights of Borland's new Turbo Pascal 4.0*

- Compiles 27,000 lines per minute
- Supports >64K programs
- Uses units for separate compilation
- Integrated development environment
- Interactive error detection/location
- Includes a command line version of the compiler

## *4.0 also*

- Saves output screen in a window
- Supports 25, 43 and 50 lines per screen
- Generates MAP files for debugging
- Has graph units including CGA, EGA, VGA, MCGA, 3270 PC, AT & T 6300 & Hercules support
- Supports extended data types (including word, long integers)
- Does smart linking
- Comes with a free revised MicroCalc spreadsheet source code

*4.0 is all yours for only $99.95*

### *Sieve (25 iterations)*

| | ***Turbo Pascal 4.0*** | ***Turbo Pascal 3.0*** |
|---|---|---|
| *Size of Executable File* | 2224 bytes | 11682 bytes |
| *Execution speed* | 9.3 seconds | 9.7 seconds |

Sieve of Eratosthenes, run on an 8MHz IBM AT

Since the source file above is too small to indicate a difference in compilation speed we compiled our GOMOKU program from Turbo Gameworks to give you a true sense of how much faster 4.0 really is!

### *Compilation of GO.PAS (1006 lines)*

| | ***Turbo Pascal 4.0*** | ***Turbo Pascal 3.0*** |
|---|---|---|
| *Compilation speed* | 2.2 seconds | 3.6 seconds |
| *Lines per minute* | 27,436 | 16,750 |

GO.PAS compiled on an 8 MHz IBM AT

***60-Day Money-Back Guarantee*****

BORLAND

BORLAND INTERNATIONAL

*For the dealer nearest you or to order call*

(800) 543-7543.

**CIRCLE NO. 101 ON READER SERVICE CARD**

## ARTICLES



## COLUMNS



## REVIEWS

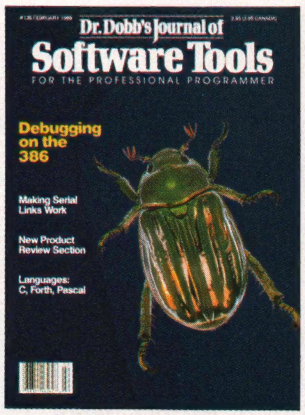


## FORUM



## PROGRAMMER'S SERVICES





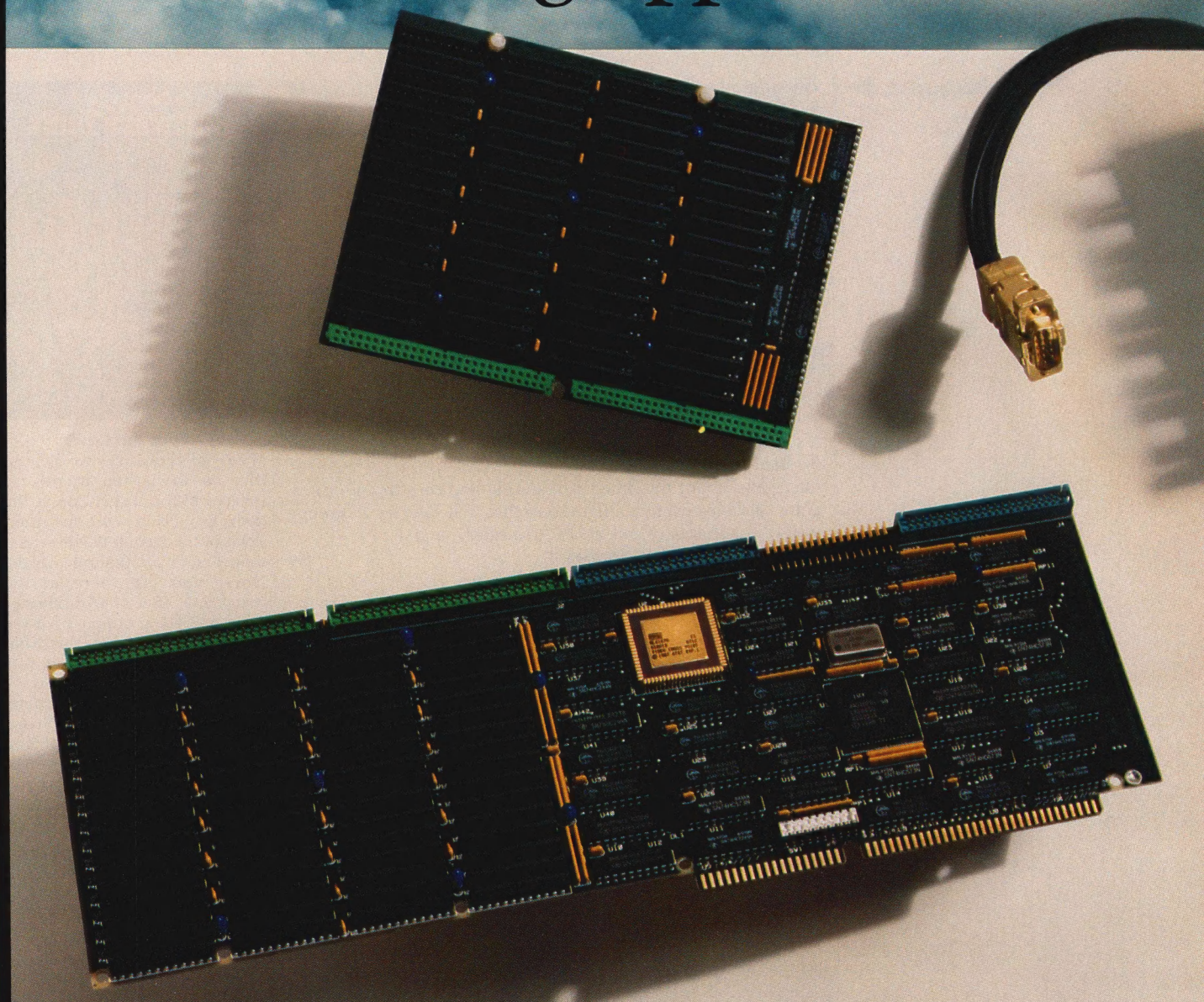

**About the Cover**

The collective Doctor would like to thank his friends in charge of the Insect Zoo at the San Francisco Zoo for their loan of the Scarab Beetle which appears so prominently on the cover and elsewhere in the magazine. After our art director had bagged a bunch of these beauties, we told him that what we really wanted to show was some fairly sophisticated ways to eliminate the pesky critters. His comments on the inappropriateness of a nuked bug on the cover, and his thorough disgust at the entire concept, forced us settle for this issue's (admittedly tasteful) cover.

**Next Issue**

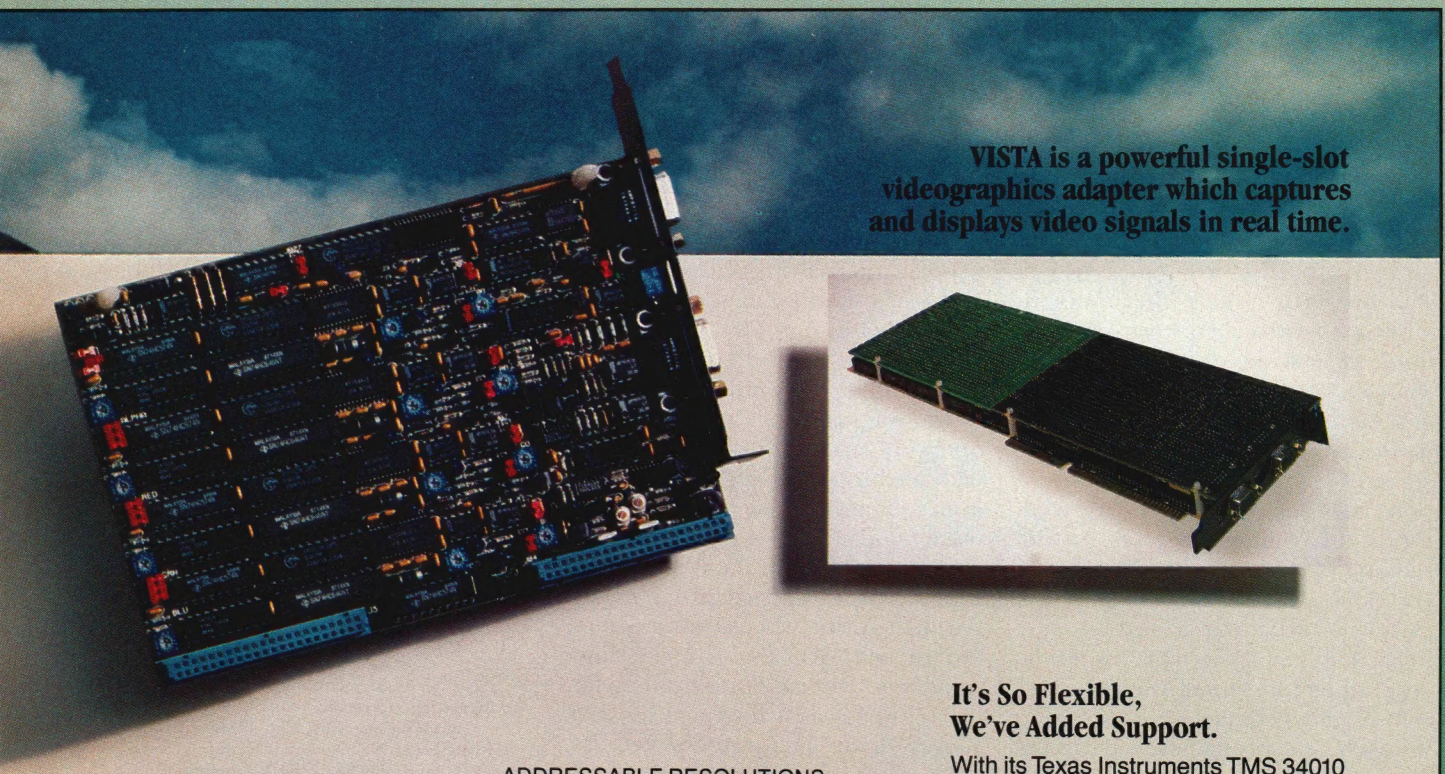
*DDJ*'s March issue welcomes back the Mac column from its temporary space crunch hiatus and features some interesting examples of the object-oriented paradigm.

# Now Taking Applications.



Take a look at the specs on VISTA™, a good look. Notice the processing, programming, and video capabilities? Now think real hard about what *you* could do with the power of VISTA and a microcomputer. Incorporate it with your system to create a digital pre-press proofing station for publishing. Design a graphics workstation which outputs both colorful hi-resolution slides and broadcast-quality animated images. Construct a CAD system which merges computer generated images with real-life backdrops for architecture, packaging or other industries. And, after you've brainstormed your way to new horizons of videographics possibilities, get your own VISTA and start working.

# Introducing VISTA™ Videographics.



**VISTA is a powerful single-slot videographics adapter which captures and displays video signals in real time.**

## Let's Get Specific.

We knew you couldn't resist seeing the facts, and frankly, our engineers wouldn't have it any other way. Here is an overview of VISTA's key features.

FEATURES:

- 4Mbytes of Video RAM on-board
- Texas Instruments' TMS 34010 GSP
- Flexible, programmable resolutions
- NTSC and PAL compatible
- Four 8-bit channels for real-time capture
- Fully integrated genlock
- Processor memory expandable in 2Mbyte increments to 12Mbytes
- Four 2K x 8-bit CMOS static RAM LUTs
- Display can be color-mapped, RGB, or a versatile combination of both
- Interlaced and non-interlaced display
- Binary and fractional programmable zoom capability, creates horizontal and vertical magnify or minify
- Smooth horizontal and vertical programmable panning, includes wrap-around and split screen
- Suggested Retail Price: $5995.

ADDRESSABLE RESOLUTIONS:

| 32 bits/pixel | 16 bits/pixel | 8 bits/pixel |
|---|---|---|
| 1024x1024 | 2048x1024 | 4096x1024 |
| 512x2048 | 1024x2048 | 2048x2048 |
| 256x4096 | 512x4096 | 1024x4096 |

CAPTURE RESOLUTIONS:*

| NTSC | PAL |
|---|---|
| (RS-170A) | (CCIR-624) |
| 756x486 | 738x576 |
| 604x486 | 590x576 |
| 504x486 | 492x576 |
| 432x486 | 422x576 |

*Resolutions are programmable; these are nominal ones for interlaced NTSC and PAL compatible.*

DISPLAY RESOLUTIONS:*

| NTSC | PAL | Interlaced | Non-Interlaced |
|---|---|---|---|
| (RS-170A) | (CCIR-624) | | |
| 1512x486 | 1476x576 | 1024x768 | 768x576 |
| 1008x486 | 984x576 | (60 Hz) | (50 Hz) |
| 756x486 | 738x576 | | |
| 604x486 | 590x576 | 768x768 | 756x486 |
| 504x486 | 492x576 | (80 Hz) | (60 Hz) |

*Resolutions are programmable; these are nominal ones.

COMPUTER REQUIREMENTS:

| | |
|---|---|
| Host Type: | IBM PC AT and 100% Compatibles, Compaq 386, Apollo DN 3000-single-slot board |
| Data Bus: | 16-bit or 8-bit (self-configuring) |
| Bus Clock: | 6MHz to 12MHz |
| Power Consumption: | 15 Watts |

## It's So Flexible, We've Added Support.

With its Texas Instruments TMS 34010 graphics processor, large quantity of video memory, and proprietary video cross-point, VISTA can be programmed for an array of powerful market-specific videographic applications. To help you maximize VISTA's potential, Truevision offers a range of C-language programming tools for developers. And when your system is market-ready, we'll support your marketing efforts with our *TRUEVISION SOFTWARE CATALOG, TRUEVISION NEWS,* and *THE PULSE.*

## We're For Higher

Resolution...Power...Flexibility... Quality. Join the many key manufacturers and developers already working with the state of the videographics art, VISTA. Call us at **800/858-TRUE** for more information on the VISTA Developer's Program. We're ready to take your application today.

TRUEVISION®

7351 Shadeland Station, Suite 100
Indianapolis, IN 46256
800/858-TRUE

TRUEVISION®

**CIRCLE NO. 103 ON READER SERVICE CARD**

# EDITORIAL

## The Doctor and Copy Rites

Every so often a letter arrives at the *DDJ* offices that demonstrates a reader's concern over the issue of software copyright. Given that we often publish both articles and source code listings that include an author's "All Rights Reserved" copyright notice, the question has been raised as to what rights (if any) the reader has. A strict reading of copyright law would lead you, after all, to the conclusion that even photocopying a section of a magazine you'd bought would be copyright infringement. Given the current litigious climate of the software industry, it's probably a good idea that we at *DDJ* at least give you a statement of our policies and attitudes concerning copyrights.

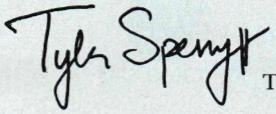
Perhaps the most prevalent confusion arises in the infamous Public Domain. The law (in this case) is pretty clear: any work placed in the public domain is available for anyone to use for any purpose. Pretty simple. Yet all too often we receive manuscripts and programs at *DDJ* with copyright notices that read something like this: "Released to the public domain. Commercial use prohibited." A legal paradox. How in the world can you enforce a right that you've given away?

So if you've been wondering why *DDJ* publishes so much code with author copyright notices instead of donations to the public domain, that paradox should be a pretty good clue. Giving away all your rights to a program is an irreversible step and while most programmers like to share their ideas and programs with others, nobody likes to feel exploited. Based on my informal polling of programmers, that's a big problem with giving away software. In the last few years we've seen a number of bulletin boards, information utilities, and bulk disk vendors make more than a little money off the distribution of public domain software.

Given the nature of the industry and *DDJ*'s tradition of sharing information with our fellow programmers, it was clear that relying on articles dealing with PD software alone wasn't going to cut it. The option of standing over our authors with a whip and trying to beat them into submissions (so to speak) just wasn't appealing.

Our solution to the problem is not final or even particularly noble. We just print a magazine every month with articles, columns, and programs. We pay the authors for the rights to print their articles and source code. The authors get the usual fame and glory, a bagful of nickels, and an occasional letter complaining that their program makes the Fastbomb C compiler blow up.

And what about the reader? What are your rights to the software? Well, despite all the lawsuits and paranoia in the software industry, the answer to that is pretty straightforward. All of us—the authors, the editors, and the publisher—agree that once you buy an issue of *DDJ* you are entitled to use that issue's source code listings in any manner you want in the privacy of your own home, for the satisfaction of your intellectual curiosity. This is our family definition of "fair use." Copying the code directly into your own programs and then distributing it is *not* what we call fair use. We call it tacky. Worse, our lawyers call it illegal. If you find yourself in a situation where you'd like to professionally use some code that appears in *DDJ*, please give us or the author a call. Odds are strong that a little politeness will be rewarded with a license that costs little (perhaps just a credit line), and everyone will feel better.



Tyler Sperry
editor

**Dr. Dobb's Journal of Software Tools**
FOR THE PROFESSIONAL PROGRAMMER

**Editorial**
**Editor-in-Chief** *Michael Swaine*
**Editor** *Tyler Sperry*
**Managing Editor** *Monica E. Berg*
**Associate Editor** *Ron Copeland*
**Assistant Editor** *Sara Noah Ruddy*
**Technical Editors** *Allen Holub, Richard Relph, Kent Porter*
**Contributing Editors** *Stan Krute, Martin Tracy*
**Copy Editor** *Rhoda Simmons*
**Art/Production**
**Director** *Larry L. Clay*
**Art Director** *Michael Hollister*
**Assoc. Art Director** *Joe Sikoryak*
**Technical Illustrator** *Barbara Mautz*
**Typesetter** *Mary E. Lopez*
**Cover Photographer** *Michael Carr*
**Circulation**
**Circulation Director** *Maureen Kaminski*
**Fulfillment Coordinator** *Francesca Martin*
**Book Marketing Mgr.** *Jane Sharninghouse*
**Subscription Supervisor** *Kathleen Shay*
**Newsstand Coordinator** *Larry Hupman*
**Circulation Assistant** *Sarah Frisbie*
**Administration**
**Vice President of Finance and Operations** *Kate Wheat*
**Business Manager** *Betty Trickett*
**Accounts Payable Supv.** *Mayda Lopez-Quintana*
**Accts. Receivable Supv.** *Laura DiLazzaro*
**Marketing/Advertising**
**Director** *Ferris Ferdon*
**Marketing Mgr.** *Michael Wiener*
**Advertising Coordinator** *Patricia Albert*
**Account Managers** *see page 129*

**Publisher**
*Peter Hutchinson*

***Dr. Dobb's Journal of Software Tools*** (USPS 307690) is published monthly by M&T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600. Second-class postage paid at Redwood City and at additional entry points. *DDJ* is published under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a nonprofit corporation.

**Article Submissions:** Send manuscripts and disk (with article and listings) to the Associate Editor.

***DDJ* on CompuServe:** Type GO DDJ

**Address Correction Request:** Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 3713, Escondido, CA 92025. **ISSN 0888-3076**

**Customer Service:** For subscription problems call: outside CA (800) 321-3333; in CA (619) 485-9623 or 566-6947. For book/software order problems call (415) 366-3600.

**Subscriptions:** $29.97 per 1 year; $56.97 for 2 years. Canada and Mexico add $27 per year airmail or $10 per year surface. All other countries add $27 per year airmail. Foreign subscriptions must be prepaid in U.S. funds drawn on a U.S. bank. For foreign subscriptions, TELEX: 752-351.

**Foreign Newsstand Distributor:** Worldwide Media Service Inc., 386 Park Ave. South, New York, NY 10016; (212) 686-1520 TELEX 620430 (WUI).



M&T Publishing Inc.
**Chairman of the Board** *Otmar Weber*
**Director** *C. F. von Quadt*
**President** *Laird Foshay*
**Vice President of Publishing** *William P. Howard*

# Aztec C

## *Power to go the distance... Whatever that distance might be*


From real time embedded applications to comprehensive commercial applications on Macintosh, IBM PC, Amiga, Atari, and others, Aztec C has earned a well-deserved reputation as an innovative, tough to beat, rock-solid C development system.

But don't just take our word for it—try it yourself. We know that the best way to understand what puts you ahead with Aztec C is to use it. That's why Aztec C systems purchased directly from Manx come with a 30-day, no questions asked, satisfaction guarantee. Call for yours today.

We can also send you information that details the special features and options of Aztec C. Plus information on support software, extended technical support options, and all of the services and specialized support that you may need when you're pushing your software to the limits and... beyond.

### MS-DOS Hosted ROM Development Systems

**Host + Target: $750  Additional Targets: $500**

**Targets:**
- **6502 family**
- **8080-8085-Z80-Z180-64180**
- **8088-8086-80186-80286/8087-80287**
- **68000-68010-68020/68881**

**Components:**
- **C compiler for host and target**
- **Assembler for host and target**
- **linker and librarian**
- **Unix utilities make, diff, grep**
- **Unix vi editor**
- **debugger**
- **download support**

**Features:**
- **Complete development system**
- **Fast development times**
- **Prototype and debug non-specific code under MS-DOS**
- **Compilers produce modifiable assembler output, support inline assembly, and will link with assembly modules**
- **Support for INTEL hex, S record, and other formats**
- **source for UNIX run time library**
- **processor dependent features**
- **source for startup**

### Aztec C Micro Systems

Aztec C is available for most microcomputers in three configurations: The Professional; The Developer; and The Commercial system. All systems are upgradable.

**Aztec C68k/Am .... Amiga**
source debugger–optional

**Aztec C68k/Mac ... Macintosh**
MPW and MAC II support

**Aztec C86 ..... MS-DOS**
source debugger • CP/M libraries

The following have special pricing and configurations. Call for details.

**Aztec C68k/At ..... Atari ST**
**Aztec C80 ..... CP/M-80**
**Aztec C65 ..... Apple II & II GS**

**Standard System ..... $199**
- C compiler
- Macro Assembler
- overlay linker with librarian
- debugger
- UNIX and other libraries
- utilities

**Developer System ..... $299**
- all Standard System features
- UNIX utilities make, diff, grep
- UNIX vi editor

**Commercial System ..... $499**
- all Developer features
- source for run time libraries
- one year of updates

MANX

C.O.D., VISA, MasterCard, American Express, wire (domestic and international), and terms are available. One and two day delivery available for all domestic and most international destinations.

**Manx Software Systems**
One Industrial Way
Eatontown, NJ 07724

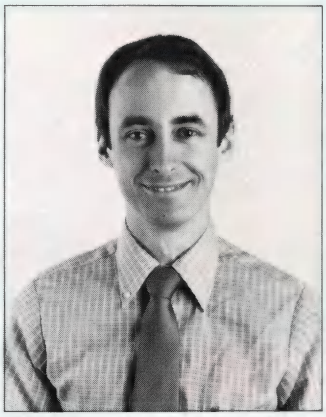
*Aztec C is available on a thirty-day money back guarantee. Call now and find out why over 50,000 users give Aztec C one of the highest user-satisfaction ratings in the industry.*

## Call 1-800-221-0440

**In NJ or outside the USA, call 201-542-2121**
**Telex: 4995812  Fax 201-542-8386**

**CIRCLE NO. 104 ON READER SERVICE CARD**

# RUNNING LIGHT



Back in November I asked what I thought was a relatively innocent question: "Are you guys really using those listings or what?" Now, following the deluge of letters that column generated, I have some answers. You use the listings.

The responses were interesting. Some of you were sure I planned to turn *DDJ* into another *Byte* by eliminating all the source code. In fact, several readers threatened to cancel their subscriptions immediately (as they had with *Byte*) if we stopped running the listings. Some readers pointed out that on-line access wasn't a practical option for all parts of the world. And a few, bless you, were charitable enough to note that I was asking for your feedback, not making a threat.

So, to follow up the topic I introduced last November: Yes, there are changes coming for *DDJ*. And no, we're not killing the source listings.

There are some repercussions from printing all that source code, however. As you must have noticed, a long listing can definitely limit the number of articles we can print in a given issue. Programs for the Mac and OS/2 environments seem especially prone to source listings that are much longer than we're used to. We'll do our best to fit things in without resorting to continuing a program through several issues of the magazine, but sometimes there's no completely satisfactory solution. (This issue, for example, we had to postpone Stan Krute's Mac column; next month it will run in its entirety.)

Listings aside, there are some changes planned for the coming months. Chief of these is a new review section coordinated by Ron Copeland. This is not an attempt to woo readers from *PC Magazine*, thank you very much, but an attempt to get more information to you about products of interest *to programmers*. We'll be printing reviews of compilers and libraries, not roundups of printers and surge protectors. Examining Room, a monthly collection of short reviews debuts this month. As with the rest of the magazine, Examining Room is designed around your needs and feedback. If there's a product you're particularly fond of then let Ron or me know by mail, e-mail, or phone.

In other review news, it seems not everyone agreed with Allen Holub's review of C compilers a few months back. Check out the letters section for some of the flames from the CompuServe forum.

On the home front, the chaos around here has abated significantly with the addition of our new managing editor, Monica Berg. Monica is a veteran of innumerable deadline battles, having just completed a tour of duty at *Unix/World* before signing on here at *DDJ*.

One last note on this issue's theme before I go. Robert Ward's *Debugging C* (Que Corp., 1986) isn't a new title, but it is still one of the best books on the subject. He covers the traditional methods and the tricks specific to debugging C as well as the use of sdb and CodeView. Regardless of how you normally work, if you're programming in C, you'll probably learn something from this book. Highly recommended.

Tyler Sperry
editor

# ARCHIVES

### Ten Years Ago Today

"One principal difficulty with newly evolving computer technology is that software generation tools generally lag corresponding hardware facilities, thus forcing the software engineer to resort to outmoded techniques to produce software systems."—*Gary A. Kildall*, DDJ, *February 1983.*

### 'This is a long song folks, and tonight its going to be even longer'

"It is often true that along with articles of major import come listings of considerable length. *Dr. Dobb's Journal* traditionally offers useful listings in their entirety, rather than just short excerpts, as illustrations for an article. Sometimes that means we do not have enough space to present the usual variety of articles, because the listings in a particular issue are so voluminous. But we are banking on the fact that the usefulness of the listings will more than compensate for those occasional times when the number of articles is fewer than normal."—*Marlin Ouverson, Editor*, DDJ, *August 1981.*

### Yes, but Is It Turbo Compatible?

"Partly as a matter of self preservation, we have become interested in the problem of standards for the Pascal language. The United States Defense Department and many large industrial corporations have recently decided to use Pascal as a base language which they would extend, and possibly alter, to create system implementation languages.... We, and many others in the Pascal User Group, are very much concerned that all this extension and alteration activity will result in Pascal going the way of BASIC for which hundreds of dialects are in common use. We believe that a chance still exists to gain consensus on a substantial family of Pascal extensions for system programming, provided that this can be brought about within the next 6 to 12 months. Unless someone does so before us, we intend to convene a summer workshop for representatives of some of the major using organizations in the hope that such a consensus can be reached."—*Kenneth L. Bowles*, DDJ, *March 1978.*

DR. DOBB'S JOURNAL of
COMPUTER
Calisthenics & Orthodontia
*Running Light Without Overbyte*

# New! Introducing Turbo C 1.5—the best optimizing compiler gets even better!

*The professional optimizing compiler for less than $100*

Turbo C® is a technically superior production-quality compiler. (Borland's equation solver, Eureka™, is written in Turbo C.) And our Turbo C 1.5 offers a new library of the highest presentation-quality graphics in the industry—the kind you'll see in Quattro,™ our new professional spreadsheet.

And spectacular graphics are just part of the brand-new features. Turbo C 1.5 enhancements also include:

- A professional-quality graphics library of over 70 functions
- A librarian that allows you to build your own object module libraries
- Context-sensitive help for the language and the library routines

Actual photograph of Turbo C graphics displayed on IBM 8514 screen.*

- Text/video functions, including windows
- 43- and 50-line mode support
- VGA, CGA, EGA, Hercules, and IBM 8514 support
- File search utility (GREP)
- Sample graphics applications
- More than 100 new functions

For professional-quality C at an affordable price, no one else comes close to Turbo C. Because no one can deliver technical superiority like Borland.

*60-Day Money-back Guarantee***

For the dealer nearest you or to order, call

(800) 543-7543

BORLAND INTERNATIONAL

BORLAND

***Minimum system requirements:*** For the IBM PS/2™ and the IBM® and Compaq® families of personal computers and all 100% compatibles. PC-DOS (MS-DOS®) 2.0 or later. 384K.

*Artwork metafile courtesy of Genigraphics® Corporation

**Customer satisfaction is our main concern; if within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.



BI 1165B

## *It's easy to upgrade to Turbo C 1.5!*

Just complete this coupon and mail it with payment before June 30, 1988. Or, call us at (800) 543-7543 and be ready to give our operators your name, credit card number, and the serial number on your Turbo C master disk.

***Turbo C 1.5 Upgrade Price*** ***$ 33.50***

CA and MA residents add sales tax _____

Shipping and handling
In US $5.00 (Outside US add $10) _____

Total amount enclosed $ _____

***Must include your Turbo C serial #*** _____

Return this coupon and the Turbo C RTL source code registration form from your Turbo C manual along with your payment by March 31, 1988 and receive your Turbo C 1.5 upgrade for free! (No phone orders please.)

***Turbo C 1.5 Runtime Library Source Code*** ***$ 150.00***

CA & MA residents add sales tax _____
Price includes shipping to all US cities.
(Outside US add $10) _____

Total amount enclosed $ _____

Please specify diskette size ☐ 5¼" ☐ 3½"

Method of Payment: ☐ VISA ☐ MC ☐ Check ☐ Bank Draft

Credit card expiration date: _____ / _____

Card # |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

Name _____

Ship Address _____

City _____ State _____

Zip _____ Phone ( ___ ) _____

***Mail coupon to:*** Turbo C 1.5 Upgrade Dept., Borland International
4585 Scotts Valley Drive, Scotts Valley, CA 95066

This offer is limited to one upgrade per valid product serial number. Not good with any other offer from Borland. Outside US make payments by bank draft payable in US dollars drawn on a US bank. CODs and purchase orders will not be accepted by Borland.

**DDJ 2/88**

CIRCLE NO. 105 ON READER SERVICE CARD

# Introducing a whole new world in C performance.

# C6.0

**WATCOM** unleashes a high-performance optimizing C compiler and development system that delivers superior results for every professional programmer and offers the fastest programming environment you can get your hands on.

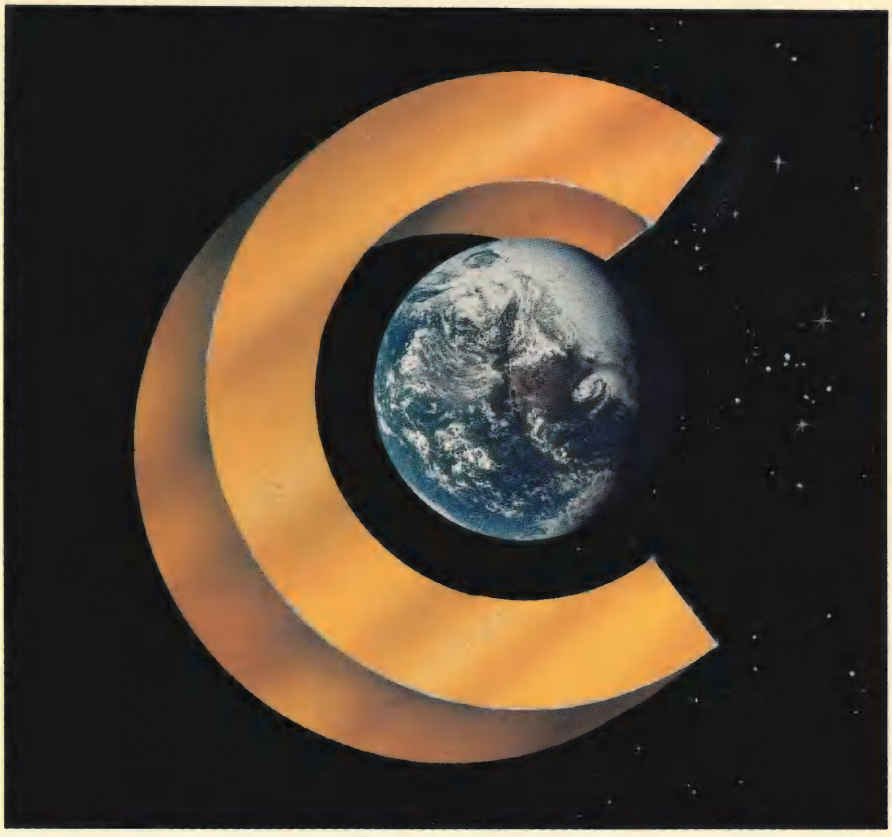
Producing the fastest, tightest code available today, WATCOM C6.0 offers major advantages in large memory models and floating-point computation.

The WATCOM C6.0 system includes WATCOM Express C (also sold separately), a seamless development environment complete with unexcelled diagnostic capabilities to maximize your productivity.

## World Class Performance

Benchmark tests pitting WATCOM C6.0 against Microsoft C5.0 and Turbo C demonstrate the optimum code size and speed your programs will achieve with WATCOM C6.0. Sophisticated register allocation, true register variables and flow analysis allow your code to run its fastest. Moreover, extensive fine-tuning is possible with numerous user options like in-line substitution of machine code for performance-critical areas.

We deliver more than performance. Flexible run-time conventions allow efficient interfacing to a wide range of libraries and language processors. WATCOM C6.0 supports small, medium, compact, large and huge memory models and the NEAR, FAR and HUGE keywords.



## Announcing the world's fastest, tightest code.

**The fastest, tightest code**
(small memory model*)

| | WATCOM C6.0 | Microsoft C5.0 | Turbo C |
|---|---|---|---|
| 25 Iterations Sieve (bytes / seconds) | 141 / 11.4 | 182 / 13.0 | 154 / 14.1 |
| Dhrystone (bytes / seconds) | 992 / 89.0 | 1246 / 98.0 | 1144 / 121.0 |

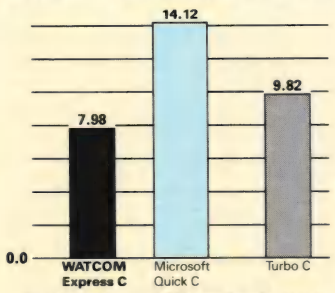
*IBM PC XT

**The fastest, tightest code**
(medium memory model*)

| | WATCOM C6.0 | Microsoft C5.0 | Turbo C |
|---|---|---|---|
| 25 Iterations Sieve (bytes / seconds) | 150 / 2.7 | 184 / 3.0 | 159 / 3.4 |
| Dhrystone (bytes / seconds) | 1001 / 18.0 | 1232 / 19.0 | 1156 / 24.0 |

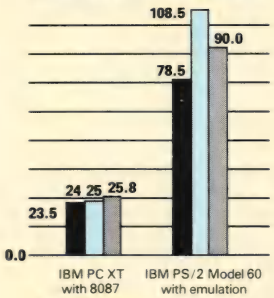
*IBM PS/2 Model 60

**Turnaround Time (compile + link/load)**




Time in seconds to compile 573 source lines plus includes, and link 7 additional OBJ files created from 1391 source lines. IBM PS/2 Model 60, small memory model. Program used: GREP.

**Floating point computation**




Time in seconds to run 25 iterations of Whetstone. Small memory model (64K code, 64K data)

## Maximum Productivity

Express C's fastest available compile turn-around assists you with syntax and prototyping. With integrated tools like an editor and a symbolic debugger, you'll be amazed at how quickly you see your results.

## The Right Tools for the Job

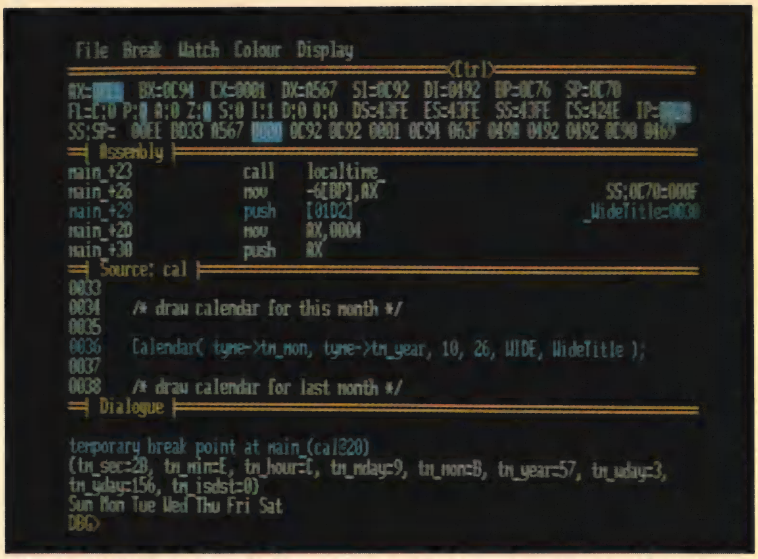
Express C, included with WATCOM C6.0, has unique error checking that quickly uncovers many common and difficult bugs in "correct" code that other compilers miss. Take advantage of the powerful integrated debugging in Express C, then use WATCOM C6.0 for even tighter code and faster execution speed. Comprehensive compile-time and run-time messages enable effective programming practices which you can extend with our fast development tools: **MAKE**, a **disassembler**, an object **librarian** and an overlay **linker**.

## Technical Support

Because we know how valuable your time is, our on-line help is always available. Our software maintenance and licensing plans will keep you up to date. With them, our expert technical support is just a phone call away. When you create a great new product, our run-time library code can be licensed without royalties or accounting. Naturally, training seminars are available at our Customer Education Center or your site.

## Superior Value

The WATCOM C6.0 system delivers superior value, returning your investment promptly by saving your valuable time. Our twenty years of computer language experience began in 1965 when our **WATFOR** authors wrote the book on high-productivity debugging compilers. The WATCOM C compiler is based on the sixth in a series of 8086-family code generators used since 1981. Today, we remain committed to delivering world class programming tools.




**With WATCOM C6.0, you can debug large applications without extended memory.**

## WATCOM C6.0

### Superlative Performance

- Full ANSI C Optimizing Compiler
- Source Editor
- Full-screen Source-level Debugger
- Full ANSI C Run-time Library
- Overlay Linker
- Object Librarian
- MAKE
- Disassembler
- **Express C**

List price: $495

Introductory Price: **$295***

## WATCOM Express C

### Unparalleled Productivity

- Full ANSI C Compiler
- Integrated Source Editor
- Integrated Debugger
- Full ANSI C Run-time Library
- Integrated Linker/Loader
- Overlay Linker
- Object Librarian
- MAKE

List price: $125

Introductory Price: **$75***

Available for IBM PCs, PS/2s and true compatibles with DOS. General availability March 1, 1988.

*Limited time introductory prices apply only to prepaid orders (VISA/MasterCard). Shipping and handling extra.

**WATCOM**

_____ I want to order **WATCOM C6.0**, regularly priced at $495 and well worth it, for just **$295**.

_____ I want to order **WATCOM Express C**, regularly priced at $125 for just **$75**.

_____ Please send me organization site licensing information.

_____ Please convince me further by sending a complete product description of WATCOM C6.0.

Name _____ Title _____

Company _____

Street _____

City _____ State _____ Zip _____

**Information needed for Credit Card orders:**

Signature _____ Tel. _____

Credit Card: Visa _____ MasterCard _____

Card # _____ Expiration Date _____

**WATCOM**
Dept. DD-02B
1430 Massachusetts Avenue
Cambridge, MA 02138

**For even faster service call:**
**1-800-265-4555**

# LETTERS


## More Listing Pros and Cons

Dear *DDJ*,

In the November 1987 "Running Light," you asked whether it was sensible to print source code listings. I think it is definitely a good policy for three reasons: some, like me, have no other access to listings; studying the printed listings is helpful in understanding a program; and those who peruse *DDJ* away from a telephone would appreciate printed listings.

You mustn't assume that everyone has ready access to a fast modem or has the money to buy one; I have limited resources and cannot afford a modem. Studying the printed listings is useful in understanding a program and its accompanying article. It would be maddening to read in the text "see line 97" without the listing at hand, especially if you're in a library. Finally, those who flip through *DDJ* in bookstores may decide to buy based on the listings. I have. In fact, I bought several issues at full retail before subscribing. So please don't stop printing source code listings in *DDJ*—they're most useful.

Dan Velting
Grand Rapids, MI 49506

Dear *DDJ*,

November 1987's Running Light provided a long overdue relief. Is it true, you are really going to spare me the monthly trip to my not-so-local newsagent? My sole reason for going there is to pick up a certain computer magazine featuring full-length source listings. If said magazine stops publishing those listings, I in turn shall stay home near the warm oven, which comes in handy with winter almost upon us.

Were I mad enough to use the twisted pair to download the source code absent from *DDJ*'s pages, the post office would charge me $8.40/minute for a phone call to the U.S. If the rare occasion should indeed find me in a spending mood, it would still be in vain, as your Bell standard modem finds itself unable to talk to my V21/V22/V23 DCE.

No, if you want me to keep on getting cold and wet feet, you will continue to print listings in *DDJ*. But who could be that cruel?

Andreas Burmester
2000 Hamburg 70
West Germany

Dear *DDJ*,

Keep the listings! Not all of us have sold our souls to IBM and MS-DOS. Not all of us live where CompuServe is a local call or can afford the charges. Not all of us have a 2,400-bps modem.

The listings are the foundation of *DDJ*. How do you read an article about a program without access to the code? If you do away with the listings, you'll be just another *Byte*—bland pap for computer illiterates.

Mad at you for even suggesting such a thing.

Donald Lashomb
Cranberry Lake, NY


*After weeks of unsuccesful debugging, Gordon resorted to a hardware breakpoint.*

## Turbo Wars

*Editor's note: A brief storm of controversy blew up on CompuServe following Allen Holub's comments on C compilers in his October 1987 "C Chest" column. A sample from the thread follows:*

Sb: C reviews 10/87
Fm: Jeff Brenton 76703,1065
To: Allen Holub 72407,3564
Dear Allen,

Is Turbo C a better product than Quick C? Probably not, but, until mere mortals can purchase and *receive* Quick C, Turbo is a better product for the "interested in C, but no corporate backing available for my education" user. As I write this note, no one I know of has received Quick C, including people who ordered it directly from Microsoft as part of the version 5.0 upgrade.

On the other hand, Turbo C, though it lacks even a symbolic debugger, has been in my hands since July, and generates better code (for the most part) than v.4.0 of Microsoft C. The fast compile times make it much more "fun" to use than any other compiler I have access to. Its earlier delivery, combined with its performance and low cost, made it possible for Borland to sell more Turbo C packages than Microsoft ever sold of MSC, just in the first months after its release! I wouldn't be surprised if Borland manages to *triple* Microsoft's C compiler sales before Quick C hits the market.

# VEDIT PLUS

# #1 PROGRAMMABLE EDITOR

## Call 1-800-45-VEDIT for FREE Fully Functional Demo Disk

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS easily satisfies the most demanding computer professional.

### Try a Dazzling Demo Yourself.

The free demo disk is fully functional—you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial—you experiment in one window while another gives instructions.

The powerful "macro" programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a "macro"—it shows that no other editor's "macro" language even comes close. And VEDIT PLUS is only 40K in size.

Go ahead. Call for your free demo today, You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

### Only VEDIT PLUS is this Flexible.

The installation lets you pick from closely emulating the keyboard layout of Word Perfect, WordStar and others. Or you can easily create your own layout and even your own editing functions. Supports any screen size—you pick screen colors and attributes.

Supports the IBM PC, XT, AT and PS/2. Also supports MultiLink, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, FlexOS (protected mode), CP/M-86 and CP/M. (Yes, we support windows on most CRT terminals, including CRTs connected to an IBM PC.) Order direct or from your dealer. $185.

**Special:** VEDIT (single file, no windows) for CP/M—$49.

- **Fully Network Compatible**
- **Call for XENIX-286 version**
- **30 Day Money-back guarantee**

### Compare Features and Speed

| | BRIEF | Norton Editor | PMATE | VEDIT PLUS |
|---|---|---|---|---|
| 'Off the cuff' macros | No | No | Yes | Yes |
| Built-in macros | Yes | No | Yes | Yes |
| Keystroke macros | Only 1 | No | No | Unlimited |
| Multiple file editing | 20+ | 2 | No | 20+ |
| Windows | 20+ | 2 | No | 20+ |
| Macro execution window | No | No | No | Yes |
| Pop-up menus | No | No | No | Yes |
| Execute DOS commands | Yes | Yes | Yes | Yes |
| Automatic processing of Compiler errors | Yes | No | No | Yes |
| "Cut and paste" buffers | 1 | 1 | 1 | 36 |
| Undo line changes | Yes | No | No | Yes |
| Paragraph justification | No | No | No | Yes |
| Convert to/from WordStar | No | No | No | Yes |
| On-line calculator | No | No | No | Yes |
| Configurable Keyboard | Hard | No | Hard | Easy |
| 43 line EGA support | Yes | No | No | Yes |
| Manual size/index | 250/No | 42/no | 469/Yes | 380/Yes |
| Benchmarks in 120K File: | | | | |
| 2000 replacements | 1:15 min | 34 sec | 1:07 min | 6 sec |
| Pattern matching search | 20 sec | Cannot | Cannot | 2 sec |
| Pattern matching replace | 2:40 min | Cannot | Cannot | 11 sec |




*Also available for TI Professional, Tandy 2000, DEC Rainbow, Wyse WY700 and others.
*Demo disk is fully functional, but does not readily write large files.

**CIRCLE NO. 107 ON READER SERVICE CARD**

# CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299, TELEX 701821

*(continued from page 12)*

*(Borland's currently estimates Turbo C sales at over 150,000.—Ed.)*

In spite of learning C in the old separate-everything environment, I still use Turbo C's integrated environment, since I haven't needed to use inline assembly yet. For the sort of quick and dirty stuff I do, I have grown to accept the clunky editor in exchange for the way it drops me into the editor and shows me "the error of my ways." I still use the DeSmet SEE editor for writing and modifying programs, however.

I also appreciate Borland's intention to deliver a "conforming" ANSI C compiler, as opposed to Microsoft's intention to "implement the important aspects" of the proposed ANSI standard.

I will not purchase Quick C by itself; I will wait until I can afford to buy the full Microsoft C v. 5.x package. After all, programming is but a hobby for me at this time, and I don't have a company buying my compilers for me. If Quick C had been available in June, like Turbo C, I *might* have bought it instead, but it wasn't, so I bought my first Borland product ever.

Sincerely,
Jeff Brenton
Woodstock, IL

Dear Jeff,
Your points about availability are well taken. Quick C was supposed to be available by now. As for "conforming" to the ANSI standard (or at least conforming as well as possible to what is, after all, a moving target), both compilers are the same. As for conforming to the de facto Unix standard for the non-ANSI functions, Microsoft is far superior to Turbo. There are problems with the Microsoft Unix support (such as no *ioctl( )*) but I'd rather have no function at all than some function that calls itself *ioctl( )* but has no relationship to the Unix function whatever, as is the case with Borland. *Signal* is another case in point.

—Allen

Dear Allen,
Your example of *ioctl( )* is a rather poor choice. By definition, such a function *must* be OS-specific; the version in Turbo provides "a direct interface to the DOS ioctl call," just as *ioctl( )* provides access to the Unix system call. Sure, it is wonderful to have a function hide the differences, but, since you are already dickering with system-specific code, what good is "portability"?

You call the Draft ANSI Standard for C a moving target, and then point to Microsoft's adoption of the "de facto Unix standard." Which one? Version 7, System III, System V, BSD? If SysV or Berkeley, which release? There are *many* differences, both major and minor, between the C compilers in these Unix systems! The Unix "standard" doesn't *have* to move—it is so broad, that you can call virtually any variation on a function's actions "Unix standard".

That was the whole purpose of X3J11—to establish a true standard, one that allowed some leeway. There are a minimum number of functions, headers and macros that must be there to be "conforming," and their use means portability to other complying compilers. Borland says they are shooting for full compliance. Microsoft says "the important parts." But who decides what is important enough to include? With all the fighting that went on over what was in and out of the standard, ANSI must think *all* of it is important!

I probably will get Quick C and Microsoft C 5.0 eventually. But Turbo C has and will continue to do well for me. Upon its arrival, DeSmet and Eco both left my hard disk. Turbo C takes up less space, generates code better than most, and most of the major bugs have been fixed in my copy.

C you around!

Sb: #Datalight Optimum-C
Fm: Dave Searles 73647,1011
To: Allen Holub 72407,3564
Allen... I just read your review of Datalight Optimum-C in the October *DDJ*. I agree with your assessment of the compiler except for one major point. Optimum-C *does* support debugging as it does produce line number information with the '-g' option. Granted that a Codeview-type debugger isn't included, but I have been very happy with my combination of Optimum-C and Periscope II. I have all the benefits of Microsoft C at less than half the price. And as far as I can tell, I'd put Periscope up against Codeview any day, especially Periscope version III. Nothing beats a *resident* debugger.... Its great when you suddenly notice an anomaly in a previously "debugged" piece of code... just hit the NMI switch and snoop around. Sorry for the sales pitch, but I thought you gave Datalight a cheap shot, even after they have given all of us a cheap shot at a really good compiler.

Fm: Bruce Kitchin 75046,1131
I really take exception to the assumption made by a number of people (usually but not always those who have little or no experience with Turbo C or in one case I remember, a person who used it for a day, ran into a glitch and gave it up and told the world that it was bad) that Borland products are for Sunday hackers. In the *DDJ* article, it was admitted that Turbo C sometimes generated better code than MSC 4.0 but not as good as MSC 5.0. For the price, I hope that MSC 5.0 generates better code, that has not been MSC's strong point up till now.

Without looking at price, I have found it difficult to see much superior about MSC 4.0 over Turbo C and I have found some ways in which Turbo C is superior (not including its greater conformance to the *draft* ANSI proposal which I assume that MSC 5.0 will catch up with, that is just a matter of release dates versus when ANSI updates the draft—a moving target). I've used both compilers with a program that contains many source files with a total line count approaching 20,000. The final code of Turbo C is smaller with both optimizing for space, I've examined a lot of code with both and find a toss up (depends on where you look), and Turbo C gave me better warnings about ambiguous code, about 20 percent of which were latent bugs that I would have had to debug when running.

**DDJ**

# The Leaders Made PVCS
# The Leading Source Code Control System.

NOW VAX/VMS & MS-DOS Versions

When it comes to maintaining their most valuable asset, the leading software publishers rely on the POLYTRON Version Control System (PVCS). From accounting firms to airlines, the leading service companies depend on PVCS to maintain the integrity of their programs. Leading manufacturing companies use PVCS to maintain their state-of-the-art software. Leading high technology companies turn to PVCS to handle configuration management for software projects that represent an investment of hundreds of thousands of dollars. The largest aerospace companies and defense contractors use PVCS to maintain integrity of projects during development and after delivery of software. Independent programmers use PVCS to improve their productivity and software quality for themselves and their clients.

## Simplify Configuration Management

When large and complex software programs are being developed on personal computers or VAX minicomputers, effective management of the revisions and versions becomes critical. PVCS simplifies this process and lets you effectively control the proliferation of code changes. We used UNIX SCCS and RCS as models. However, our own experience, and the input of hundreds of programmers and managers has enabled us to significantly improve upon these models.

## PVCS provides many powerful functions including:

- Storage & Retrieval of multiple revisions of text.
- Maintenance of a complete history of changes.
- Maintenance of separate lines of development using branching.
- Merging simultaneous changes.
- Resolution of Access Conflicts.
- Modules can be retrieved by their own revision number, system version name, or specified date.
- Uses "reverse deltas" to rebuild a prior version making PVCS the fastest version control system over the project life cycle.
- Projects already under development or in the maintenance stage can be easily put under the control of PVCS.

## Manages Development On Local Area Networks

Programming teams using Local Area Networks depend on PVCS to help the managers and team members work together. In fact, Novell and 3Com themselves depend on PVCS to manage the versions of their own network software products.

## Supports MS-DOS and VAX/VMS Development

Now, companies that develop software on VAX systems running VMS can also use PVCS. And since the VMS and MS-DOS versions of PVCS use the same "logfile" format, you can easily develop software on PCs and maintain the code on the VAX or vice versa. The menu-driven, screen-oriented interface (and optional command-driven interface) makes it easy for programmers and librarians or administrators to use PVCS on a PC or VAX or both systems.

## PVCS Maintains System Integrity

PVCS prevents corruption of code that could ordinarily result from security breaks, user carelessness or malfunctions. The levels of security can be tailored to meet the needs of your project.

## PVCS & PolyMake Work Together

PolyMake, the leading MS-DOS make utility, is now available for the VMS operating system. This allows you to write makefiles that will function in both PC and VAX environments. Additionally, PolyMake reads time & date stamps of PVCS archives for fast, accurate program rebuilding.

## PVCS and PolyMake Maintain Source Code Written In Any Language.

## The Preferred Version Control System

The customers listed below are just a few of the innovative leaders that have made PVCS the leading version control program for personal computers.

Alcoa Aluminum
Arthur Anderson
AT&T
Ashton-Tate
Bank of America
Bell Labs
Bendix
Boeing
CIGNA
Citibank
3Com
Colonial Penn
Commerce Clearing House
Control Data Corp.
Corvus
CXI
Digital Equipment Corp.
Deloitte Haskins + Sells
Diebold
Dow
Dunn & Bradstreet
EDS
Educational Testing Service
E-Systems
Equitable Life
Federal Express
First Boston
Ford
Fox Software
Fujitsu
GTE
Hardees
Hewlett-Packard
Honeywell
Hughes Aircraft
IBM
Industrial Networking
Intel
ISC Aerospace
IVAC
Javelin
Lattice
Lawrence Livermore
Lotus
McData Corp.
McDonnell Douglas
Mead Data Central
MIT Lincoln Labs
Nastec
Novell
NCR Technologies
Pitney Bowes
Plexus Computers
Price Waterhouse
ROLM
Rockwell International
Safeco
Sears
Security Pacific
Sperry
Software Publishing
Spacelabs
Standard Oil
Standard & Poors
Tandem
Tektronix
Telex
Texas Instruments
Touche Ross
Unisys
United Airlines
United Parcel Service
United Technologies
U.S. West
Westinghouse Electronics
Xerox

Only PVCS meets the needs of independent programmers and corporations. Once you standardize on PVCS, the archives used to track and monitor changes are interchangeable between any PVCS product. You will receive full credit for your initial purchase if you upgrade to a higher-priced MS-DOS version of PVCS.

**Personal PVCS** — Offers most of the power and flexibility of Corporate PVCS, but excludes the features necessary for multiple-programmer projects.

**Corporate PVCS** — Offers additional features to maintain source code of very large and complex projects that may involve multiple programmers. Includes multi-level branching to effectively maintain code when programs evolve on multiple paths (e.g. new versions for different host systems, or a new program based on an existing program).

**Network PVCS** — Extends Corporate PVCS for use on Networks. File locking and security levels can be tailored for each project.

**PVCS for VAX systems** — Requires VMS. Uses the same interface and archive format as MS-DOS version. Supports branching and offers file locking and other security features for multiple-programmer projects.

| | MS-DOS* | VMS | | |
|---|---|---|---|---|
| | PC/XT/AT | Micro VAX II | VAX 7xx | VAX 8xxx |
| Personal PVCS | $149 | | | |
| Corporate PVCS | $395 | | | |
| Network PVCS | $995** | $4,950 | $9,500 | $10,500+ |
| PolyMake | $149 | | | |
| Network PolyMake | $447** | $1,250 | $2,375 | $2,500+ |

*Compatible with MS-DOS 2.0 through 3.3. Compatible with the IBM PC/XT/AT & other MS-DOS PCs.

**5 Station LAN License. Call for pricing on larger Networks.

TO ORDER:
VISA/MC 1-800-547-4000.
Dept. No. 355
Oregon & Outside USA call (503) 645-1150.
Send Checks, P.O.s to: POLYTRON Corporation, 1815 NW 169th Place, Suite 2110, Beaverton, OR 97006.

POLYTRON
*High Quality Software Since 1982*

CIRCLE 108 ON READER SERVICE CARD

# Texas Instruments has system developers need.

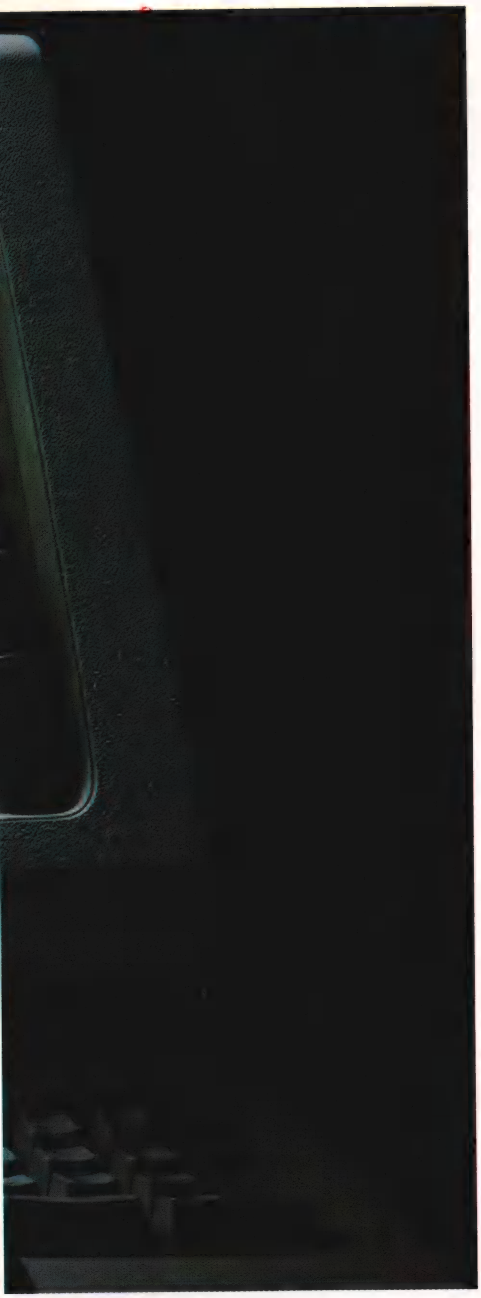

"Personal Consultant™ Plus…offers a very fine expert system development and delivery tool that already has a proven record with end-users."

— *Susan Shepard, AI Expert*

**Personal Consultant Plus 3.0 Standard Features**
- Frames, rules, meta rules and procedures
- Forward/backward chaining
- Confidence factors
- Regression testing and rule tracing
- End-user explanation facilities
- Graphics image capture and display
- Interfaces to dBase™, Lotus 1-2-3™, DOS files, .EXE or .COM programs, "C"
- Complete LISP development environment
- 2-megabyte expanded/extended memory support
- Mouse support
- Context sensitive help
- "Getting Started" tutorial-style manual

**Personal Consultant Images**
- Optional add-on package to PC Plus (3.0)
- Allows integration of "active images" into

# what serious expert Power tools.



knowledge bases
- Interactive dials, gauges, forms and selection images
- Multiple images can be combined on same screen
- "Getting Started" tutorial-style manual

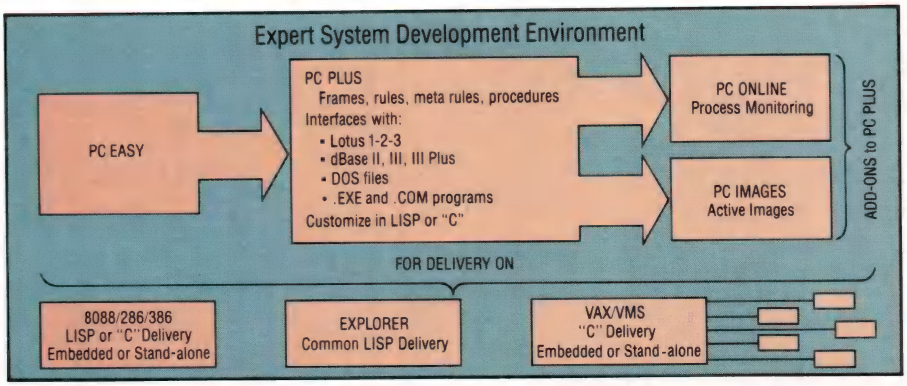
**Personal Consultant Online**
- Optional add-on package for PC Plus (3.0)
- ONLINE expert systems that interact directly with process data
- Multiple interfaces to data acquisition and analysis programs
- Knowledge base synchronization with process data
- Functions for historical and predicted trends
- Special user interface/reporting capabilities
- "Getting Started" tutorial-style manual

Among all the expert system development tools available for personal computers today, none deliver the power and flexibility of TI's Personal Consultant series.

Personal Consultant Easy is ideal for getting started, and is upwardly compatible with the higher functionality of PC Plus. For experienced developers, Personal Consultant Plus and its optional add-on enhancements, Online and Images, were designed to help solve a broader range of complex problems.

package helps deliver expertise that is "online all the time."

**Application delivery as flexible as the tools themselves.**
Delivery can be in LISP for flexibility, or "C"* for maximum speed and portability. Our "C" options support either stand-alone or "embedded" knowledge bases. Options are available for DOS-based PCs, TI's Explorer, and DEC's VAX™ line of multi-user minis running under VMS™.




**Personal Consultant Plus. Full power for an affordable price.**
At $2,950, PC Plus has proven to be one of the richest and most flexible problem-solving tools available for the development of complex knowledge-based systems. Designed to take advantage of today's more powerful 286/386 DOS-based computers, or TI's Explorer™ Symbolic Processing System, the new 3.0 version of PC Plus provides powerful standard features and a continuing growth path with the addition of either PC Images or PC Online, or both.

**Personal Consultant Images. Picture an expert system with interactive graphics.**
At $495, PC Images enables developers to create knowledge-based applications that incorporate complex graphical "active images." User-interactive dials, gauges, forms and selection images provide a more exciting visual data input and output style.

**Personal Consultant Online. The expert system as part of the process.**
At $995, PC Online allows the developer to design expert systems which interact directly with process data, as opposed to input from a human operator. Designed for intelligent process monitoring applications, this optional

***"Texas Instruments has done more than any other company to educate people about AI, to popularize it, and to make useful AI tools available at reasonable prices."***
*— Jim Seymour, PC Magazine.*

Technical support, training courses and Knowledge Engineering Services are available for the Personal Consultant products. If you have a question about any of our expert system power tools, we have the answer.

**Pick up the phone and gain a powerful advantage.**
Call **1-800-527-3500** for technical overviews of our products and a PC Plus case histories brochure which details how our power tools are being put to work today.

36106

Personal Consultant and Explorer are trademarks of Texas Instruments Incorporated.
dBase is a trademark of Ashton-Tate.
Lotus 1-2-3 is a trademark of Lotus Development Corp.
VAX and VMS are trademarks of Digital Equipment Corporation.
*Available 4Q 1987.

TEXAS INSTRUMENTS

# Debugging with the 80386

*Notes on real mode debugging with the 386*

by Franklin Grossman

Intel's 80386 processor has received a great deal of praise for its improvements over the earlier 80286. Much has been written on the excellent speed and (essentially) flat address space of the 386. In developing software, however, there are other concerns that are just as important as operating speed, even though fast development is often a critical factor. In this article I'll discuss some of the advantages of the 386 processor from the perspective of software debugging.

When Intel designed the 80386, it included a new feature: hardware support for debugging. Probably the most important addition of the new debugging support was the inclusion of breakpoint registers, which allow breakpoints on memory reads, writes, and instruction fetches. Although breakpoint registers are powerful, they do have limitations. By using a few other features of the 80386, our company was able to create a software debugger that contains most of the features found in hardware-assisted debuggers. In addition to breakpoint registers, our product, Soft-ICE, uses several protected mode features, such as virtual 8086 mode, paging, I/O privilege level, and breakpoint registers, to add real-time hardware-level breakpoints and other features found only in hardware-assisted debuggers to existing DOS software debuggers. This article describes how these 80386 features work and how our debugger uses them.

### Virtual-Machine Capability

All 80x86 real-address-mode software debuggers cause side effects to the program environment. These side effects are caused because the debuggers use memory, interrupt vectors ([*INT 1*], [*INT 2*], and [*INT 3*]) and DOS or BIOS for I/O. Software debuggers are also at risk of being overwritten or affected in other ways by the target program. For this reason, many hardware-assisted debuggers load the debugging code into write-protected memory on an option card. Even this solution can cause side effects because that memory is mapped into the lower Mbyte that is visible to the 80x86 processor in real-address mode, and this address space (such as *C0000H* or *D0000H*) is also used by other adapter cards. By using the 80386 in virtual 8086 mode, though, it is possible to write a debugger that surrounds the DOS environment in a virtual machine without any of the side effects mentioned earlier.

The 80386 provides a virtual 8086 capability intended for use by protected mode operating systems. This feature was necessary because 80386 protected mode is not backward compatible with 8086 executable programs. The 8086 virtual-machine capability is implemented in such a way that a protected mode operating system can allow multiple virtual 8086 tasks that are controlled by the operating system kernel, and so the operating system and other native tasks are isolated from ill-behaved DOS programs. The operating system cannot be overwritten by the DOS program and has complete control over interrupts, I/O, and the memory map.

Several operating systems commercially available today use the virtual 8086 mode of the 80386. These include FlexOS (DRI), PC-MOS/386 (Software Link), VPix (Phoenix Technologies and Interactive Systems), and Windows/

*Franklin Grossman is the president of Nu-Mega Technologies and one of the principal designers of the Soft-ICE debugger.*

*(continued from page 18)*

386 (Microsoft).

If you think of the model described earlier but replace the operating system with a debugger, you get some interesting benefits. The debugger can control the 8086 environment without affecting it or being affected by it, and the debugger code does not run in the virtual 8086 task and therefore is not visible to DOS or to DOS programs. To implement a debugger based on this model, the debugger must have many features of an operating system, including a complete I/O system—the debugger cannot rely on DOS or BIOS for I/O.

A protected mode debugger has more control over the virtual 8086 task than is possible with a conventional software debugger. Interrupts are controlled because all interrupts go through the protected mode interrupt table to the protected debugger. (It is the responsibility of the protected debugger to generate the interrupt in the 8086 virtual machine if necessary.) I/O is controlled because the protected debugger has control over which *IN* and *OUT* instructions are passed through to the hardware and which cause exceptions (80386 exceptions are very similar to 8086-style interrupts). The memory map is completely controlled by the 80386 paging mechanism. Using paging, the amount of memory given to the virtual machine can be varied in 4K increments up to 1 Mbyte. In most instances 640K is the right number. Memory pages can also be marked as "not present," causing an exception if a program running in the 80386 virtual machine accesses that memory page and so giving the debugger control over access of memory regions.

This article describes how protected mode features can be used to provide sophisticated debugger breakpoints. The breakpoints are generally implemented by gaining control of the processor when an exception occurs. At this point a debugger window can be popped up or control can be given to a conventional DOS software debugger. This process is described in detail later.

### Breakpoint on I/O

A useful feature in debugging device drivers is having breakpoints on *IN* and *OUT* instructions. *IN* and *OUT* instructions execute under control of the protected debugger. The 80386 gives the operating system the ability to trap on accesses to any I/O address, a capability that was included to allow the operating system to "virtualize" the I/O of an ill-behaved DOS program. An example of this would be capturing bytes output to the parallel port in a printer driver application, where the protected mode operating system could send these bytes to a print spooler.

The 80386 allows the protected mode operating system to control the virtual machine's access to I/O ports through a bit mask. The bit mask contains a separate bit for each 8086 I/O port. If the bit is clear, the 80386 lets the *IN* or *OUT* instruction execute normally. If the bit is set, the 80386 generates an exception that is handled by the protected mode operating system. I/O addresses range from 0 to 65,535, so a complete bit mask takes 65,536 bits, or 8K of memory. If the protected mode operating system provides a bit mask of less than 8K, then any accesses to I/O ports that are not covered by the bit mask cause an exception.

Our protected debugger takes the place of the protected mode operating system, using the I/O bit mask to provide breakpoints on *IN* and *OUT* instructions. To set an I/O breakpoint, the debugger sets the bit that corresponds to the specified I/O address. When the target program running in the virtual 8086 task accesses that I/O address, an 80386 exception occurs. An 80386 exception is similar to a software interrupt. At this point the debugger has control, but it does not know if an *IN*, *INS*, *OUT*, or *OUTS* instruction caused the exception.

The protected debugger's exception handler can look at the actual instruction that caused the exception to determine the instruction type. More specific breakpoints are possible by comparing the actual value being output with a predefined value. In the case of an input, the instruction can be single-stepped and the value in the *AL* or *AX* register compared with a predefined value. If these specific criteria are not met, the debugger can give control back to the virtual 8086 task.

One advantage of using the 80386 virtual-machine features to cause I/O breakpoints is that 80386 exceptions occur instantaneously. This may not seem like a revolutionary statement, but the builders of hardware-assisted debuggers and in-circuit emulators have been confronted with this problem for years. As microprocessors become more pipelined, it is difficult to cause an instantaneous breakpoint—for example, most hardware-assisted debuggers generate an NMI (nonmaskable interrupt) when the breakpoint conditions are met. Because the 80386 prefetches and predecodes several instructions before the actual target instruction is executed, the 80386 has actually executed several instructions before it recognizes the NMI.

### Breakpoint on Interrupt

When debugging, it is often useful to set a breakpoint on a hardware or software interrupt. You may, for example, want to run a program until it makes a DOS call to read the version number. You could set a breakpoint for *INT 21* with *AH = 30*. Interrupt breakpoints are a natural for protected debuggers.

In our protected debugger, all interrupts go through the protected mode interrupt table that is under complete control of the debugger. It is the responsibility of the protected debugger to get the address from the 8086 virtual mode task's interrupt vector table at *0:0* and transfer control to that address. With a little additional qualification code that compares the interrupt number with a value previously input by the user, you have breakpoint-on-interrupt capability. This method works equally well with hardware or software interrupts.

### An End to Hung Programs

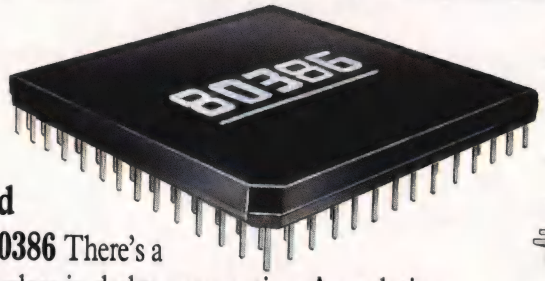
Another debugging feature that can be implemented using the virtual machine is the ability to pop your debugger up at any time, even if interrupts are disabled or masked off. Often, when a program is hung, it is


# The New Standard Bearer.

THE SOFTWARE LINK

CIRCLE NO. 109 ON READER SERVICE CARD

# A Number of Reasons A Number

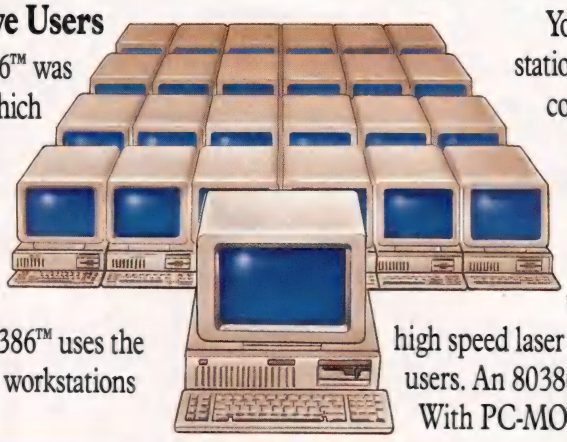



**1. Designed for the 80386** There's a revolution taking place in desktop computing. A revolution that's been launched by a square wafer of silicon known as the 80386 microprocessor chip. It puts minicomputer potential at PC users' fingertips. It's a fact that virtually every leading PC manufacturer has built a "box" around this chip. And it's a fact that the "New Operating System" will, supposedly, even run on it. But, it's also a fact that *their* system wasn't designed for the 80386. Ours is. And it's called PC-MOS/386™.


**2. PC and PS/2 Compatible** In designing PC-MOS, we knew our first priority was to exploit the minicomputer capabilities of 80386-based PCs & PS/2s. But we went further, and developed a system which would be fully compatible with the millions of existing PCs, PC ATs, and PC-compatibles. Power without sacrifice. You'd expect nothing less from the new standard bearer.

**3. One, Five, Up to Twenty-five Users** From the beginning, PC-MOS/386™ was designed as a versatile operating system which could support twenty-five users as easily as it supports one. The system comes in single, five, and 25-user modules, so you're able to start with what you need and expand when you're ready.

In a multi-user setting, PC-MOS/386™ uses the computing power of the host PC to drive workstations linked to standard RS-232 ports.



You can choose from a variety of workstations. Mix and match dumb terminals costing under $500 each with PCs and PS/2s running our terminal emulation software.

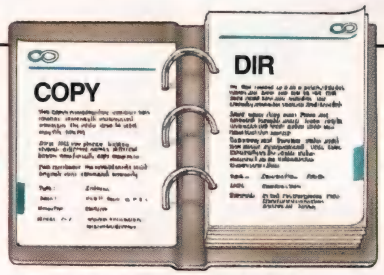
All of the host's resources can be shared. Programs, data, hard disks, tape backup units & printers (including high speed laser printers) are suddenly available to all users. An 80386-PC has minicomputer potential. With PC-MOS/386™ you can "mini" your micro.

**4. Thousands of DOS Programs** PC-MOS/386™ gives you the best of the past, and the best for your future. Which means that while PC-MOS/386™ totally replaces your old DOS, you won't have to replace the programs you've spent a lot of time learning.

And it all happens so effortlessly. You'll continue to reap the benefits of your favorite DOS programs, while entering a new arena of power.

Think of it! Programs like dBASE III, WordPerfect, Lotus 1-2-3 and Symphony, WordStar, MultiMate...literally thousands of DOS programs–all compatible and multi-user available.

**5. Familiar Commands Like DIR and COPY**




Just as you don't have to learn a whole new array of software to take advantage of PC-MOS/386™ neither do you have to learn an entirely new set of commands.

Instead, the system builds on the knowledge you already have. "COPY" still copies files, and "DIR" still gives you a directory listing. As you might expect, we didn't stop there. There's a wealth of features that have strengthened the commands you know, making them more powerful and easier to use.

# of Users Will Choose PC-MOS/386.™

## 6. Concurrently Supports Virtual 8086 and 80386 32-Bit Mode


80386-based PCs & PS/2s are dual-personality computers. To run DOS programs, they act as PCs with a 640K memory limit. But to take advantage of their minicomputer capacity, they operate in true 80386 mode which lets them address up to four gigabytes of memory. PC-MOS enables the 80386-host and its workstations to independently switch between these modes–making DOS compatibility and 80386 power simultaneously possible.

## 7. Multi-Tasking

While it's true you could look elsewhere for multi-tasking, why would you want to? The *other* multi-tasking operating system is not now, nor is it planned to be, multi-user. It won't even run multiple DOS applications in multi-tasking mode.


Now consider PC-MOS/386.™ At the touch of a key, you can switch between up to 25 different tasks. And if you have workstations connected to a host, they get multi-tasking, too. Finally...a system that won't hold you back.

## 8. File/Record Locking and Security

When you decide to implement either a network or a multi-user system, there's a two-fold problem which must be solved: protecting your work from accidental misuse and securing it from intentional theft. PC-MOS/386™ solves both aspects of this problem. Password protected security allows you to assign file, directory, and task access to each user. Plus, files and records are locked using either PC-MOS' proprietary system or NETBIOS emulation.

## 9. Remote Access

It's been said that information is power...which makes PC-MOS/386™ a deadly weapon to your competition. Imagine on-the-road salespeople being able to file call reports and access your latest inventory data. Picture executives being able to access your corporate database from across the country, or around the world–giving them the information they need, when they need it.

Visualize branch offices tapping time-critical data with nothing more than a modem and a workstation. Working at a home office in the evening or over the weekend suddenly gets awfully productive. And that makes good business sense. The kind of sense you can't afford to be without.

## 10. The Price...

As you evaluate operating systems, ask yourself if it's reasons you're considering...or rhyme. Ask if you're getting a system for tomorrow, or one that was made for yesterday. See if you're being forced to buy new hardware because of *their* software. And consider this.

Only one operating system in the world can give you the raw power, features, and functionality that you demand. Its name is PC-MOS/386.™ And it's immediately available in one, five and 25-user versions starting at $195.




| For the dealer nearest you, CALL: 800/451-LINK | In Georgia: 404/441-2580 | International/OEM Sales: 404/263-1006 | Resellers/VARs: 404/448-5465 |
|---|---|---|---|

3577 Parkway Lane, Atlanta, GA 30092 Telex 4996147 SWLINK FAX 404/263-6474

The Software Link/Canada CALL: 800/387-0453

**DEALER INQUIRIES INVITED**


**PC-MOS/386™**

MODULAR OPERATING SYSTEM

**THE SOFTWARE LINK**

*(continued from page 20)*

useful to pop into your debugger and poke around. Conventionally, this is done with an external button that is linked to an option card that causes an NMI. The conventional method often has problems because so many option cards, including most popular multivideo cards, use NMIs.

A protected mode debugger managing a virtual 8086 environment can actually provide this breakout capability through a key sequence. The specific 80386 feature that is used to provide the breakout capability is called privilege levels. To understand privilege levels you must understand a little bit about 80386 protection. The 80386 has four different protection levels, numbered level 0 (highest privilege) through level 3 (lowest privilege). The four different privilege levels can be used for four layers of differing trust levels.

In our debugger application, we need only two levels: level 0 and level 3. The debugger—as you'd expect—runs at level 0, while the virtual 8086 task runs at level 3. A program running at level 0 has complete access to all 80386 protected features. A level 3 program in contrast, cannot access 80386 control registers and other 80386 features. The ability to access I/O ports can be set (by the operating system or in this case by the protected debugger) at any level.

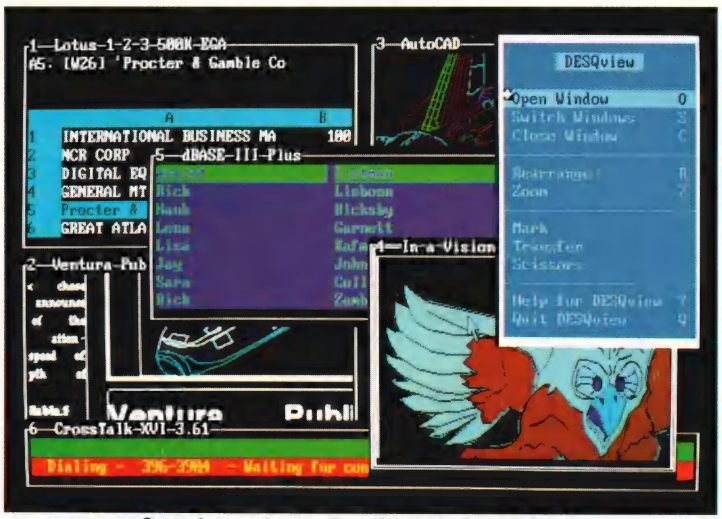
If the privilege level is set to 3, the target program running in the virtual 8086 task has some additional restrictions. Certain instructions that have to do with interrupts cause exceptions. These instructions are *STI*, *CLI*, *LOCK*, *INT*, *PUSHF*, *POPF*, and *IRET*. If you are an assembly-language programmer, you may have noticed that most of these instructions affect the processor interrupt flag. If the 80386 had two interrupt flags—one for the virtual machine and one for the native environment—it would not be necessary to monitor these instructions. Because the 80386 does not keep track of the state of the interrupt flag separately for the virtual machine, the protected debugger must monitor all instructions that cause a change in the state of the interrupt flag.

By getting control when any of these instructions are executed by the target program, it is possible to "virtualize" the interrupt system. In the case of the breakout feature, the only concern is to handle the keyboard interrupt. When the target program disables interrupts, the debugger must continue to get keystroke interrupts. The keyboard interrupt must be handled by the debugger but cannot be passed through to the virtual 8086 task. When keyboard interrupts occur, the protected debugger must monitor the keystrokes looking for a key sequence. If the sequence is found, the debugger is popped up. The tricky part is making sure all keyboard activity meant for the target environment is passed through accurately.

For hard-core systems types, it is worth mentioning that accesses to the interrupt controller mask register must be monitored as well. This is necessary in the cases in which interrupts are masked at the interrupt controller instead of at the processor interrupt flag.

# How to tell the difference between DESQview™ 2.0 and any other environment.

Selecting DESQview, the environment of choice, can give you the productivity and power you crave, without the loss of your old programs and hardware. If you like your existing programs, want to use them together, transfer data between them, print, sort, communicate with or process-in-background, yet still have the need to keep in place your favorite PC(8088, 8086, 80286 or 80386), DESQview is the "proven true" multitasking, multi-windowing environment for you. Best of all, DESQview 2.0 is here now, with all the money saving, time saving, and productivity features that others can only promise for the all-too-distant future.




*One picture is worth a thousand promises.*

And with DESQview's new graphics enhancements for Hercules, CGA, EGA, and VGA, Version 2.0 still offers the same award winning and pioneering features for programs that earned DESQview its leadership, only now you can also run desktop publishing programs, CAD programs, even GEM-™, Topview-™, and Microsoft Windows-™ specific programs. In some cases you'll add as little as 10–40K to your system overhead. Now you can have multi-tasking, multi-windowing, break the 640K habit too and still get an auto dialer, macros, menus for DOS and, for advanced users, a new complete application programmer's interface capability. No wonder that over the years, and especially in recent months, DESQview, and now DESQview 2.0 have earned extravagant praise from some of the most respected magazines in the industry.

*"Product of the Year"* by readers vote in InfoWorld.

*"Best PC Environment"* by popular vote at Comdex Fall in PC Tech Journal's "System Builder" Contest.

*"—I wouldn't want to run an IBM or compatible computer without DESQview"*—InfoWorld, Michael Miller.

*"A colossus among windowing environments"... "will run almost anything"*—PC Week, Marvin Bryan.

*"Windows, promises, but DESQview delivers"*—MICRO-TIMES, Birell Walsh.

No other environment has consistently pioneered features, openness, and productivity. See for yourself. Send in the coupon. The possibilities are endless with DESQview 2.0.

Attention Programmers: For more information about Quarterdeck's API, and future 386 program extensions, call us today.

SYSTEM REQUIREMENTS
IBM Personal Computer and 100% compatibles (with 8086, 8088, 80286 or 80386 processors) with monochrome or color display; IBM Personal System/2 • Memory: 640K recommended; for DESQview itself 0-145K • Expanded Memory (Optional): expanded memory boards compatible with the Intel AboveBoard; enhanced expanded memory boards compatible with the AST RAMpage • Disk: Two diskette drives or one diskette drive and a hard disk • Graphics Card (Optional): Hercules, IBM Color/Graphics (CGA), IBM Enhanced Graphics (EGA), IBM Personal System/2 Advanced Graphics (VGA) • Mouse (Optional): Mouse Systems, Microsoft and compatibles • Modem for Auto-Dialer (Optional): Hayes or Compatible • Operating System: PC-DOS 2.0-3.3; MS-DOS2.0-3.2 • Software: Most PC-DOS and MS-DOS application programs; programs specific to TopView 1.1, GEM 1.1 and Microsoft Windows 1.03 • Media: DESQview 2.0 is available on either 5¼" or 3½" floppy diskettes


Rush me DESQview 2.0! Today! DDJ 2/88

| No. of Copies | Media 3½"/5¼" | Product | Retail Price ea. | Total |
|---|---|---|---|---|
| | | DESQview 2.0 | $129.95 | $ |
| | Shipping & Handling | USA | $ 5.00 | |
| | | Outside USA | $ 10.00 | $ |
| | | Sales Tax (CA residents) | 6.5% | $ |
| | | | Amount Enclosed | $ |

Payment: ☐ Visa ☐ MC ☐ AMEX ☐ Check

Credit Card: Valid Since ____/____ Expiration ____/____

Card Number: _____

Credit Card Name _____

Shipping Address _____

City _____ State _____ Zip _____ Telephone _____

Mail to: Quarterdeck Office Systems, 150 Pico Boulevard, Santa Monica, CA 90405.
NOTE: If you own DESQview call us for a special upgrade offer, or send in your DESQview registration card. AST Special Edition users included.

DESQview™ Quarterdeck

PC MAGAZINE EDITORS' CHOICE NOV. 24, 1987

Quarterdeck Office Systems • 150 Pico Boulevard, Santa Monica, CA 90405 • (213) 392-9851

*(continued from page 24)*

### *Memory Range Breakpoints*

The next debugging feature I'll discuss is memory range breakpoints. Memory range breakpoints are especially useful when an errant program is overwriting a portion of memory. By trapping on the write, you can find the actual code that has gone astray.

To implement memory range breakpoints, the protected debugger uses an 80386 protected mode feature called paging. The paging mechanism in the 80386 was intended for providing demand-page virtual memory in a protected mode operating system. Memory is divided into 4K pages, and the operating system can mark each page as present or not present in the 80386 page tables. If a program is executing and it enters a page that is not present, an exception occurs. It is the responsibility of the operating system paging exception handler to load the actual contents of that page from a mass storage device. Paging takes advantage of the fact that most programs tend to spend most of their execution time in a few concentrated areas.

Again, our debugger takes on the role of an operating system to manage the paging mechanism. When the user specifies a memory range, the debugger must first determine which pages the range covers. The debugger marks these pages not present in the 80386 page tables, and control is given back to the user program. If the user program accesses one of the pages marked not present, an exception occurs, and the debugger's exception handler gets control at this point. The 80386 passes the exception handler the address that was accessed in control register 2 (*CR2*). In most cases the specified memory range does not start exactly on a 4K page boundary. When an exception occurs, the debugger must compare the actual address accessed with the actual range specified. If the access was within the 4K page but not within the specified range, control is given back to the target program.

This boundary condition problem can cause performance of the target program to degrade visibly in some instances, although in most cases the performance hit is negligible. One instance in which the performance drop is noticeable is if the top of the program's stack is in the page but not within the range. Even with this performance hit, range breakpoints using paging are still thousands of times faster than the software simulation technique that some debuggers use.

A refinement of range breakpoints is possible. The 80386 paging mechanism allows pages to be read-protected and write-protected, which gives the debugger the ability to allow memory range breakpoints on read, write, or read/write accesses.

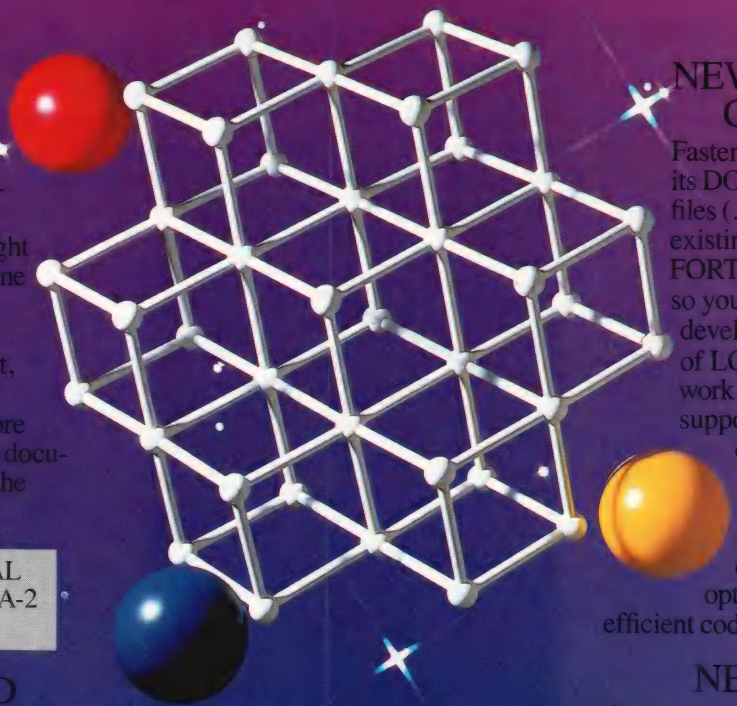
Range breakpoints occur immediately when using the 80386 paging mechanism. The instruction that caused the breakpoint to occur is also restartable, so once the memory is present, the program can continue without missing an instruction. This gives the protected debugger the same advantage over hardware debuggers that it has with I/O breakpoints; breakpoints are not affected by the 80386 instruction pipelining.

# SOFTWARE ENGINEERING COMES OF AGE.

## ANNOUNCING LOGITECH MODULA-2 VERSION 3.0

Modula-2 is the language of choice for modern software engineering, and LOGITECH Modula-2 is the most powerful implementation available for the PC. The right language and the right tools have come together in one superior product. Whether you're working on a small program or a complex project, with LOGITECH Modula-2 Version 3.0 you can write more reliable, maintainable, better documented code in a fraction of the time at a fraction of the cost.

**FREE** TURBO PASCAL TO LOGITECH MODULA-2 TRANSLATOR

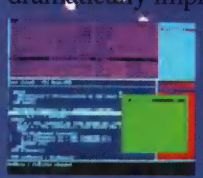


### NEW, IMPROVED COMPILER

Faster and more flexible. Now its DOS linker compatible object files (.OBJ) can be linked with existing libraries in C, PASCAL, FORTRAN and ASSEMBLER — so you can build on previous development and put the power of LOGITECH Modula-2 to work for you right now. Fully supports Wirth's latest language definition, including LONGINT and LONGSET, which provides large set support including SET of CHAR. Provides optimization for tighter, more efficient code generation.

### NEW, IMPROVED DEBUGGERS

Time gained with a fast compiler can be lost at debug time without the right debugging tools. With the powerful Logitech Modula-2 Debuggers you can debug your code *fast*, and dramatically improve your overall project throughput. The Post Mortem Debugger analyzes the status of a program after it has terminated while the dynamic, Run Time Debugger monitors the execution of a program with user-defined break points. With their new, mouse based, multiple-window user interface these powerful debugging tools are a pleasure to use.



### NEW, INTELLIGENT LINKER

Links only those routines from a particular module that you need, so you eliminate unreferenced routines and produce smaller, more compact executable files.

### NEW EDITOR

Our new, mouse based editor is fully integrated, easy to learn, fast and easy to use, and very customizable. Its multiple, overlapping windows and color support make it easy to manage parts of one file or several files on the screen at one time. You'll love using it — with or without a mouse.

Call for information about our VAX/VMS version, Site License, University Discounts, Dealer & Distributor pricing.

To place an order call toll-free:
**800-231-7717**
In California:
**800-552-8885**

☐ LOGITECH Modula-2 V. 3.0 Compiler Pack — **$99**
Compiler in overlay and fully linked form, Linkable Library, Post Mortem Debugger, Point Editor

☐ LOGITECH Modula-2 V. 3.0 Toolkit — **$169**
Library sources, Linker, Run Time Debugger, MAKE, Decoder, Version, XRef, Formatter

☐ LOGITECH Modula-2 V. 3.0 Development System — **$249**
Compiler Pack plus Toolkit

☐ Turbo Pascal to Modula-2 Translator — **FREE**
With Compiler Pack or Development System

☐ Window Package — **$49**
Build true windowing into your Modula-2 code.

☐ Upgrade Package
Call LOGITECH for information or to receive an order form.

Add $6.50 for shipping and handling. California residents add applicable sales tax. Prices valid in U.S. only. Total Enclosed $_____

☐ VISA ☐ MasterCard ☐ Check Enclosed

Card Number _____ Expiration Date _____

Signature _____

Name _____

Address _____

City _____ State _____

Zip _____ Phone _____

LOGITECH

**LOGITECH, Inc.**
6505 Kaiser Drive, Fremont, CA 94555
Tel: 415-795-8500

**In Europe:** LOGITECH, Switzerland
Tel: 41-21-87-9656 Telex 458 217 Tech Ch

**In the United Kingdom:** LOGITECH, U.K.
Tel: 44908-368071 Fax: 44908-71751

*(continued from page 26)*

### *Breakpoint Registers*

Another 80386 feature that fits in with our protected debugger is the previously mentioned breakpoint registers. Breakpoint registers were designed specifically for debugging purposes. They can be used in real-address mode and can be implemented very easily in a simple terminate-and-stay-resident (TSR) program or directly in a user program.

The 80386 includes four breakpoint registers that can be used to set breakpoints for a byte, word, or double-word. These breakpoints can be on write, read/write, or execute accesses. The four breakpoint registers are named *DR0*, *DR1*, *DR2*, and *DR3*. Each breakpoint register holds the 32-bit linear address of the byte, word, or double-word of interest. The address must be on a word or double-word boundary for those respective data types.

There are two additional registers: *DR6* and *DR7*. *DR6* is a status register that is read after the breakpoint has occurred to determine which of the four breakpoints was triggered. *DR7* is a control register that is written to specify the parameters for a particular debug register—for example, write-only on a double-word. *DR7* also contains two enable bits that must be set to activate the breakpoint.

When a breakpoint goes off, an 80386 exception occurs. The exception handler must read the status register to determine which of the four breakpoints was triggered.

As with the I/O and range breakpoints, you can easily extend the capabilities of the debug registers by adding qualifying code to your exception handler. Other features that our Soft-ICE debugger provides by additional qualification code are breakpoint on read-only and comparison with a data value. A read-only breakpoint can be implemented by decoding the instruction that caused the exception to determine if it is a memory read. If not, control is returned to the target program.

Using breakpoint registers to perform breakpoints on execution has an advantage over the conventional *INT 3* approach. Software debuggers for the 80x86 place an *INT 3* at the address of the desired execute breakpoint. An *INT 3* is used because it is a special single-byte instruction (most interrupts are 2-byte instructions) included in all 80x86 processors specifically for providing breakpoint capability. The *INT 3* approach has the disadvantage that it cannot work in ROM; the breakpoint registers work fine in ROM code.

### *Triggering Your Favorite Software Debugger*

Optimally, the protected debugger should provide all the necessary debugging commands, such as dump, unassemble, modify, single-step, display registers, and so on. Many people, however, are addicted to their favorite language-specific debugger. For these people, it would be nice to extend the capabilities of their existing debugger by adding the features that are possible with the protected debugger. This is possible. The conventional debugger runs in the 8086 virtual machine and the protected debugger runs in protected mode. The DOS environment is still affected by your conventional software debugger, but you can add additional breakpoint capability, such as breakpoint on memory range or I/O ports.

Our Soft-ICE debugger provides both methods. For users who need a complete systems debugger, we provide all the necessary debugging commands. For those who wish to extend the capability of their existing software debuggers, we have a pop-up window that allows them to set sophisticated breakpoints that will trigger their software debugger.

Triggering the conventional software debugger is possible by understanding a little about the way most 80x86 software debuggers work. Most software debuggers use the *INT 3* approach described earlier to provide breakpoints on execution. They also use the 80x86 *INT 1* single-step mechanism. All members of the 8086 line have the capability to single-step the next instruction. The debugger must set a special processor flag called the trap flag, and when the trap flag is set, an *INT 1* occurs after every instruction is executed.

By relying on the *INT 1* vector or the *INT 3* vector pointing to the conventional debugger's breakpoint-handling routines, we can generate *INT1*s or *INT 3*s to wake up a conventional debugger. Most debuggers handle unsolicited *INT* 1s or *INT 3*s beautifully; however, a few will not. The third approach uses NMI. Most debuggers have a method of handling unsolicited breakpoints through the NMI mechanism—for example, CodeView provides this with a command-line switch. They provide this capability to break out of hung programs when the user presses an external button. This button is wired through the PC bus to the NMI pin. By taking advantage of these three conventional breakpoint mechanisms, we can wake up almost any conventional debugger.

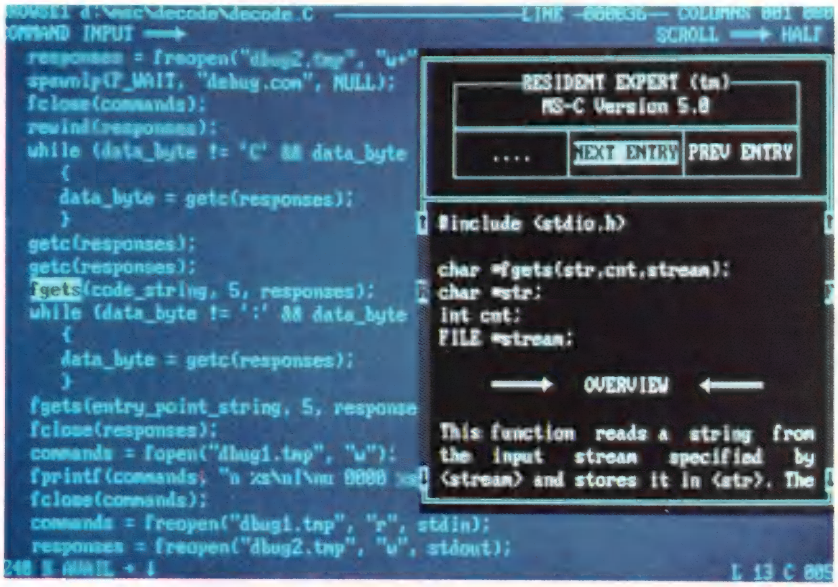
A side effect of waking up a debugger unknowingly is a problem with reentrancy. Many debuggers enable interrupts or use DOS for I/O. If you wake the debugger up while the processor is in an interrupt routine, or within MS-DOS or the ROM BIOS, the debugger will fail. Waking up a nonreentrant debugger is still useful for application-level debugging. Many debuggers are mostly reentrant, and you can wake these up at any time; Periscope I and II are examples of these.

### *Conclusion*

To build a complete protected debugger requires several additional components. These include "virtualizing" the video display system to save and restore the screen at any time and likewise the keyboard so users can debug keyboard device drivers. The purpose of this article was to describe debugging features, so I haven't gone into those details. By creatively applying 80386 protected mode operating system features in a real-address mode debugger, you can provide most of the features found in a hardware-assisted debugger with the convenience of a software debugger.

**DDJ**

# RESIDENT EXPERT Pop-up Reference Guides...




# Use One of Ours or Build Your Own!

### THE POP-UP REFERENCE REVOLUTION BEGINS

How much development time could you save if you never had to open another PC language or technical reference manual again? What if you could just point at a compiler keyword, assembly instruction, or function name *on your screen* and with a keystroke have complete, authoritative information about language syntax, operands, parameters, examples, and much more?

### INTRODUCING THE RESIDENT EXPERT SYSTEM

A growing library of comprehensive, disk resident reference guides about the PC *and* your favorite PC languages. All available instantly through our unique memory resident pop-up access system.

### VIRTUALLY EVERYTHING YOU NEED TO KNOW

Each of our *Compiler Reference Guides* contains virtually everything you need to know to program with your *preferred implementation* of your favorite language. Language syntax, all library functions, compiler directives, and error codes are thoroughly documented.

Our *PC Programmer's Reference Guide* documents every PC (and AT) processor instruction and every BIOS and DOS service interrupt. You'll also find tables of keyboard codes, line drawing, ASCII, and IBM character sets, and much more.

### THE SPECIALIST'S LIBRARY

Your compiler is unique. That's why our reference guides are *specialized*...each one designed for a particular vendor's language implementation.

***NEW!!***

## RESIDENT EXPERT Compiler

Make your own Reference Guides

### QUICK DRAW ACCESS SYSTEM

*Point-and-shoot*...just place the cursor over any term on your screen. Chances are we've got it fully detailed in one of our data bases.

*Fully cross indexed*...if the instruction or library function you're using isn't quite right, our related topics cross index can help you find a better one.

*Multiple volumes on line*...you can have one or a dozen of our pop-up reference guides on line...a complete library available instantly.

### THE INFORMATION YOU NEED...WHERE YOU NEED IT

Our pop-up shell varies its size and shape dynamically, only taking as much space on your screen as it needs and it *never* covers your working area. You can see your work and our reference data at the *same* time.

RESIDENT EXPERT Shell (*) ..... $19.95
(with PC-DOS/MS-DOS Reference Guide)

RESIDENT EXPERT Compiler..... $39.95
(create your own Reference Guides!)

RESIDENT EXPERT Reference Guides
Borland Turbo C (v1.0) ..... $19.95
Borland Turbo Pascal (v4.0) ..... 19.95
Borland Turbo Prolog (v1.1) ..... 19.95
Lattice C (v3.2) ..... 39.95
Mark Williams Let's C (v4.0) ..... 19.95
Microsoft C (v5.0) ..... 39.95
Microsoft Quick C (v1.0) ..... 19.95
PC Programmer's Reference Guide ... $39.95

*The RESIDENT EXPERT Shell is required to access and display all Reference Guides....

# SantaRita

For the location of your nearest Santa Rita Software dealer, or to order direct, call us at 1-214-727-9217. We'd like to hear from you.

**Santa Rita Software**
**1000 E. 14th Street, Suite 365**
**Plano, Texas 75074**

# The RESIDENT EXPERT System

# A Serial Protocol Analyzer Program

## *Debugging the bit stream with Turbo Pascal*

by Craig A. Lindley

This article describes the implementation of an RS-232 serial protocol analyzer (SPA) program hosted on an IBM PC computer. It utilizes the multitasking kernel I presented in the July 1987 issue of *DDJ*. The software provided in this article converts a PC into a tool that can be used to monitor most RS-232 connections.

The program was developed as an alternative to spending $5,000–20,000 on a dedicated serial protocol analyzer device. Don't misunderstand, this program (in its present form) doesn't have half the features and functions provided by a dedicated protocol tester, but it performs very well when simply displaying RS-232 data—the function a dedicated analyzer is used for 95 percent of the time. What's even better is that the SPA software is available, so you can tailor it to your specific applications.

This program will find use in many software development labs, computer/data-processing centers, and just about anywhere else that RS-232 devices are used. In this day and age of corporate frugality, buying or borrowing an IBM PC to run an application of this type is probably a lot easier than trying to justify the cost of a dedicated protocol analyzer. Also, you can't run Lotus 1-2-3 or word-processing software on your dedicated protocol analyzer when it isn't being used for its designed function.

*Craig A. Lindley, 6 Sutherland Pl., Manitou Springs, CO 80829. Craig has been working with real-time operating systems for many years. He has worked at the Jet Propulsion Laboratory on the implementation of part of the real-time system to be used on the upcoming Galileo spacecraft and is currently employed at Rolm Corp. as a software engineer involved in real-time telephony control.*

### What Does an SPA Do?

An SPA provides a visual picture of data flowing between two serial devices connected via an RS-232 interface. In addition, it provides a method of monitoring the RS-232 handshake lines (RTS, CTS, DTR, DSR, and so on) in a manner not unlike the two-color LED arrangement used on RS-232 breakout boxes. Figuring out what a serial interface is doing (or is not doing) without a device of this sort is extremely difficult.

The SPA will find use in two major areas—first, in software labs to assist in the development of RS-232 data protocols for new products or devices, and second, in data communication users environments in figuring out why two supposedly compatible serial devices are not talking to each other. I'll give an example of each application.

Consider the development of a terminal emulator program. The SPA could be used to verify that key codes programmed on a special function key were indeed being sent out of the serial port when the special function key was pressed. The SPA could also be used to verify that the terminal program could really generate a break condition on its port lines. Finally, the SPA could be used to verify the terminal emulator's implementation of the Xmodem file transfer protocol.

In a data communications environment, consider the case of a serial printer loosing some of the characters sent to it. The SPA could be used to determine that the printer was sending the XOFF software handshake command but that the source of the serial data was not responding to it by stopping its transmission of data.

These are two of the many possible applications of the SPA program. In a general sense, the SPA can take the guesswork out of serial device interfacing and can be considered a serial interface debugger. Any aids in the debugging process will equate to increased productivity and reduced frustration.

### What Is Required

The following is the minimum equipment required for use of the SPA program:

1. An IBM PC, PC/XT, or PC AT (or

compatible) with at least 256K RAM. An AT is required for data rates greater than 4,800 baud.
2. Two serial ports: COM1 configured at address 3F8H using interrupt IRQ4 and COM2 configured at address 2F8H using interrupt IRQ3.
3. The SPA executable program (SPA.COM).

If you would like to experiment with the SPA code, these additional items are required:

1. 256K of additional memory for compiling the program in memory and executing it.
2. Turbo Pascal, Version 3.0 or later, for the IBM PC.
3. The following source code files:
- SPA.PAS (Listing One, page 44)—the main program file
- MENU.PAS (Listing Two, page 66)—the menuing subsystem
- SERIAL.PAS (Listing Three, page 80)—the serial port handlers
- MULTI.PAS—the multitasking kernel presented in the July 1987 issue of *DDJ*, with slight modification.

Personally, I think the SPA is a useful program. In addition, if you examine the software components that make up the SPA, you'll find the following components are useful in themselves:

1. A generalized multitasking kernel written in Turbo Pascal.
2. A generalized 1-2-3-like menuing system that can be converted for use in another application simply by changing its database.
3. A BIOS-independent, interrupt-driven, serial interface package supporting both the COM1 and COM2 ports. This, too, is written in Turbo Pascal.

The moral of this story is, if you can't use the whole program, maybe you can use the pieces.

### *Connecting the SPA for Use*

Two methods exist for connecting the SPA to the RS-232 interface to be monitored. The first I call the passive monitoring connection and is shown in Figure 1, right.

This diagram shows the minimum connections necessary to use the SPA to monitor an RS-232 connection. In this case the SFA does not pass the data through. It merely monitors the data sent between both serial devices.

The additional monitor lines can be connected to any of the serial lines you wish to monitor on the SPAs' screen. The signal connected to: pin 5 will show up as CTS; pin 6 will show up as DSR; pin 8 will show up as CD; pin 22 will show up as RI. With this connection method the PC, which is hosting the SPA program, does not sit between the two ends of the RS-232 cable. The two endpoint serial devices are connected just as they would be without the SPA attached. The SPA is effectively connected in parallel to

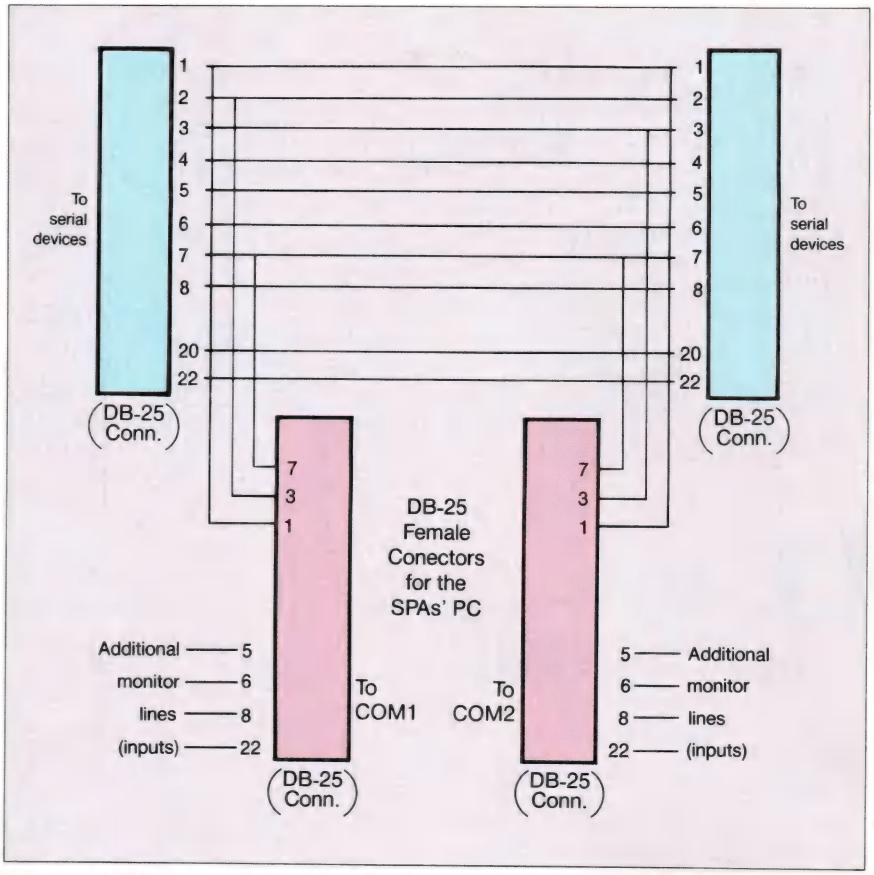



***Figure 1:*** *The passive monitor connection*

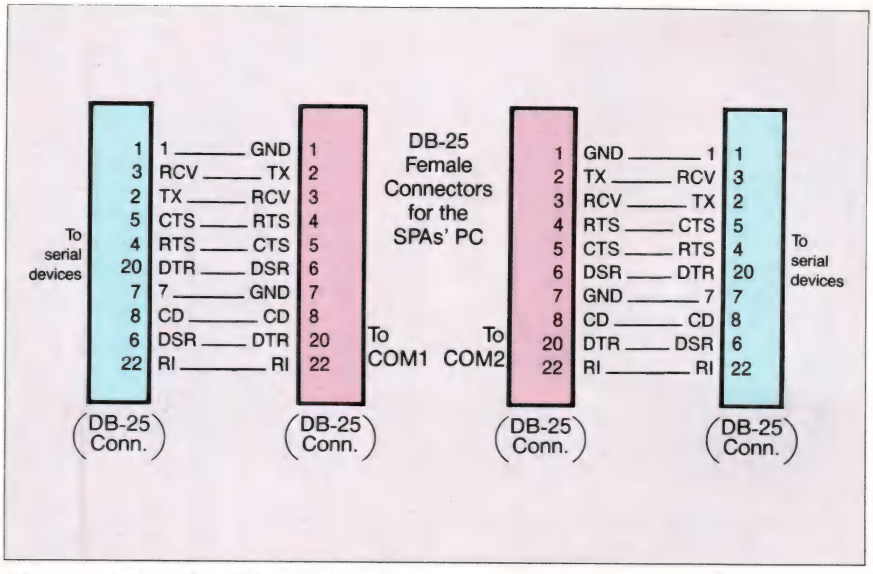



***Figure 2:*** *Pass through monitoring*

the serial data path. The data being transmitted by the devices on both ends of the serial interface is routed to the receive data inputs of COM ports 1 and 2 of the SPA's PC. Also, any of the four available inputs to the COM ports (CTS, DSR, CD, or RI) can be tied to any of the lines of the serial interface being monitored to allow the SPA to display their states visually to the user.

The second connection method I call the pass through connection. Figure 2, page 31, shows the cabling required for this connection method. With this connection, the SPA program sits between both serial devices. All data and handshake line states generated and received by both serial endpoint devices pass through (and can therefore be processed by) the SPA program. This is the mode for which the current SPA software is optimized. It allows the maximum flexibility for future SPA program functionality.

The CD and RI lines may require additional connections if the serial devices being monitored require that these signals are actively driven.

```
= COM1 ========= Serial Protocol Analyzer  Ver: 1.0 ========== COM2 =
|In Fifo:  0%   OutFifo 0% |       By      |In Fifo:  0% Out Fifo: 0% |
|Stat- BRK:- FE:- PE:- OR:-|Craig Lindley|Stat- BRK:- FE:- PE:- OR:-|
|In - CD:M RI:S DSR:S CTS:S|Display      |In - CD:M RI:S DSR:S CTS:S|
|Out - DTR:S    RTS:S      |Fifo: 3%     |Out - DTR:S    RTS:S      |
=============================================================================
```

***Figure 3:*** *The main SPA display screen*

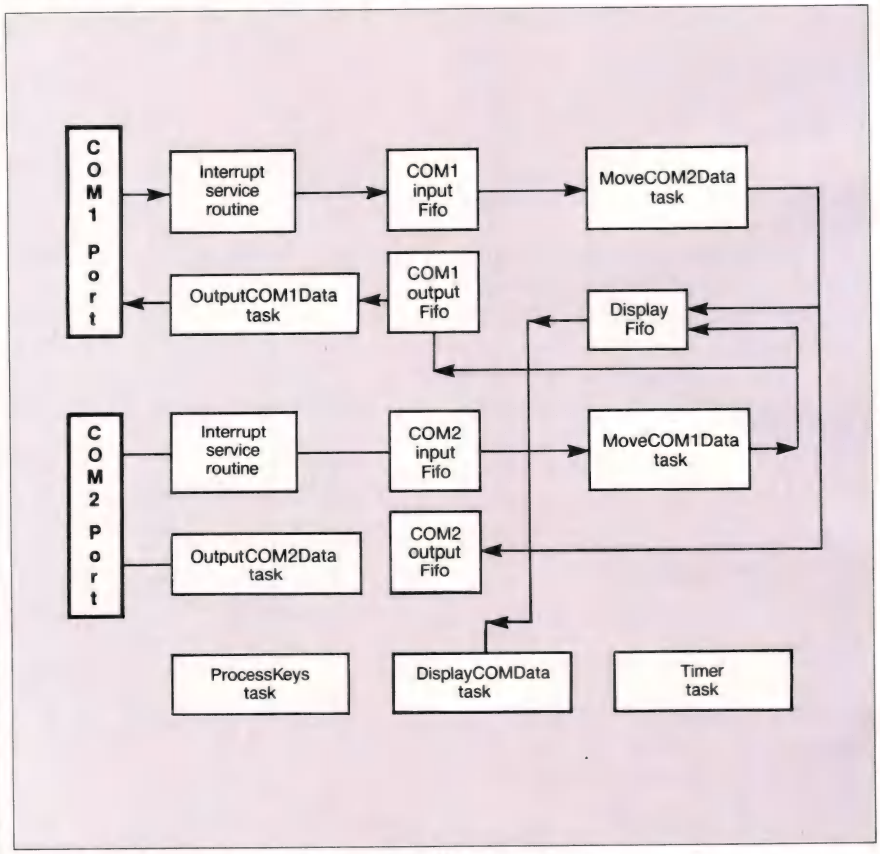



***Figure 4:*** *The block diagram of the SPA program.*

The PC serial hardware does not have the driver outputs necessary to drive these signals as it does the RTS and DTS signals. The CD and RI lines from one side could be connected to the same signal pins on the other side.

### How to Use the SPA

After connecting the SPA's PC to the serial devices using one of the two connection methods, you must execute the SPA program. The SPA program is normally compiled into an executable .COM file called SPA.COM. To execute it just type *SPA* at the DOS prompt.

The first thing you'll notice is that the SPA program verifies the presence of two serial ports (COM1 and COM2 at the proper addresses). Unless two serial ports are found, the program will halt with an error message.

If the hardware verification is successful, the operator is presented with the main display screen, similiar to that shown in Figure 3, page 32. As you can see, this screen presents a lot of information about both the COM1 and the COM2 ports. I'll now describe each item:

*In Fifo*, *Out Fifo*, and *Display Fifo*: These numbers, expressed in percentages, indicate how full the various FIFOs used by the SPA program are. See the block diagram in Figure 4, page 32, for an explanation of the function of each of these FIFOs.

Note: These percentages are not updated in real time, only every half second. They are shown to provide a feel for how well the SPA is processing the serial data. It is quite possible, however, to have a FIFO overflow occur even though the numbers suggest there is still considerable room in the FIFO.

*Stat—BRK*, *FE*, *PE*, and *OR*: These are the various status and error conditions of the COM ports in the PC. *BRK* indicates a break condition on the line, *FE* indicates a framing error, *PE* indicates a parity error, and *OR* indicates an overrun error has occurred. See the IBM technical reference manual for an explanation of these conditions. The no-error condition is indicated with a dash whereas the error condition is indicated by an *E* or *B* in the case of a break.

*In—CD*, *RI*, *DSR*, and *CTS*: These are the various input lines into the COM ports. Their states are continually monitored by the SPA program, and any changes are shown on the display. Their states are either *M* for marking or *S* for spacing.

*Out—DTR* and *RTS*: These are the two output lines from the COM ports. Their states are also either *M* for marking or *S* for spacing. Note: The SPA program routes what is received on one COM port's DSR line to the other COM port's DTR line, effectivley passing the state through the PC. The same is true of the CTS and RTS lines. This handshake line pass through happens in both directions.

The information line at the bottom of the display informs the user that:

1. If the Esc key is pressed, the menu system will be entered. The highest-level menu is shown in Figure 5, page 34.
2. If the space bar is pressed while data is being displayed, the display will pause until the Enter key is pressed to restart it.
3. If COM1 data is being displayed

```
=====================================================================
                  Serial Protocol Analyzer Menu
                    -- Main Menu Selection --
 ------
 |Quit|  Parameters   Display   Trigger   Format   Control   End
 ------
  Exit Analyzer Menus
=====================================================================
```

***Figure 5:*** *Main menu of the SPA program*

```
Main Menu - Transition Chars = QPDTFCE
 (0,0,0)
   |
Quit    Parameters   Display   Trigger   Format   Control   End
(1,0,0)  (2,0,0)     (3,0,0)   (4,0,0)   (5,0,0)  (6,0,0)   (7,0,0)
ccode=1 Parameters - Transition Chars = QBSWP
 (2,0,0)
   |
Quit    Baud Rate   Stop Bits   Wrd Length   Parity
(2,1,0)  (2,2,0)     (2,3,0)     (2,4,0)     (2,5,0)
Baud Rate - Transition Chars = Q361249
 (2,2,0)
   |
Quit      300      600     1200     2400     4800     9600
(2,2,1) (2,2,2) (2,2,3) (2,2,4) (2,2,5) (2,2,6) (2,2,7)
       ccode=2 ccode=3 ccode=4 ccode=5 ccode=6 ccode=7
Stop Bits - Transition Chars = Q12
 (2,3,0)
   |
Quit        1         2
(2,3,1)  (2,3,2)  (2,3,3)
.        ccode=8  ccode=9
```

***Figure 6:*** *The A tree menu structure (partial)*

**menu_entry = RECORD**

**title** —Item or submenu name. This string with a maximum length of ten characters is displayed on the selector line and can be selected by pressing a key corresponding to the first character of its name.

**desc** —Description of item or submenu up to 40 characters in length. Explanatory text describing the item or submenu.

**chars** —A string of upper case characters which can be used to select any item on this menu level. This string usually contains the first character of each items title.

**index** —This entry contains the length of the **"chars"** string above.

**ccode** —Only non zero at A tree leaf nodes. It is the command code to be processed by the procedure **ProcessCmd** which is associated with this menu entry.

**END;**

***Figure 7:*** *A Menu Entry Record*

*(continued from page 34)*

(the data received at the COM1 port), it will be shown in normal video. If COM2 data is being displayed, it will be in reverse video to allow an easy visual distinction.

To set up the SPA program for use, you must enter the menu system using the Esc key. Normally, the setup consists of selecting the proper baud rate, parity, and word length of the serial data stream to be monitored. The defaults chosen for all other setup parameters are adequate for an initial data display. Once the setup is complete, the endpoint serial devices should be enabled to start data transmission. If all is well, the data being passed back and forth between the serial devices should be shown on the SPA's display.

The menu system has the capability of altering the operation of the SPA program. In all, there are 35 possible alterable parameters that determine how the SPA operates. The procedure *ProcessCmd* in the file SPA.PAS (Listing One) shows exactly how each of the menu options is processed. A quick trip through the menuing system will acquaint you with flexibility of the SPA program. Note that data will still be acquired and displayed by the SPA program while you are using the menus. That is the beauty of a multitasking system.

The more important features of the menu system are:

1. Under the Parameters menu, you set the baud rate, parity, and number of stop bits with which the data should be interpreted. The program supports rates from 300 to 9,600; five through eight data bits; odd, even, or no parity; and one or two stop bits. If the serial parameters are set incorrectly, the data displayed on the SPA's screen will be garbage.
2. Under the Display menu, you set whether the data from COM1, COM2, or both is displayed on the screen.
3. Under the Trigger menu, you set up the SPA's triggering capability. Here you input the channel (COM1 or COM2) to trigger from, the single-byte trigger data pattern, and the trigger mode (display data until trigger or display data after trigger). You can also stop (wait for) the trigger in this submenu. Note: After the trigger parameters are set, the trigger must be enabled before it goes into effect.
4. The Format submenu is where the format of the displayed data is set. The options are:
   - ASCII data without handshake (default)
   - ASCII data with handshake (handshake shown in hex)
   - Hex data without handshake
   - Hex data with handshake

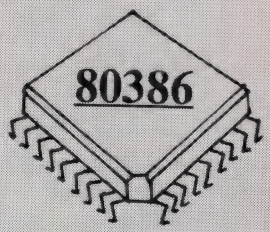
When data is displayed with the handshake option, the port input lines that were acquired when the data was acquired and certain COM port error conditions will be displayed. The information is bit encoded into the displayed hex byte as shown in Table 1, page 37.

The final option in the Format menu is simply called SPACE. This is a toggle that determines whether the data displayed on the screen is separated or not—in other words, whether a space will be inserted between each displayed data item. If a space is being inserted and this menu option is selected, the spac-

# SERIOUS DEBUGGING AT A REASONABLE PRICE

NEW

# Soft-ICE™




## All the speed and power of a hardware-assisted debugger at a software price - $386

## Features

**Real-time break points on:**
- Memory locations
- Memory ranges
- Execution
- I/O ports
- Interrupts (hardware and software)

**Dual monitor support**

**System memory map**

**Regain control with a keystroke**
Even with the following code:

```
CLI
MOV   AL,0FFH
OUT   21H,AL
JMP   $
```

**and much, much more**

## How Soft-ICE works

Soft-ICE unleashes the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment. Soft-ICE uses 80386 protected mode features, such as paging, I/O privilege level, and break point registers, to add real-time hardware-level breakpoints to your existing DOS debugger. To use Soft-ICE you simply pop the Soft-ICE window up through a key sequence, set your hard break points, then return to your soft debugger. As the target program is executing, Soft-ICE recognizes when the breakpoint conditions have been reached and gives control back to your soft debugger. And this is all done at full 80386 speed! Soft-ICE can also be used in stand-alone mode. This comes in handy if you are debugging loadable device drivers, interrupt handlers, or terminate and stay resident programs. All of the standard debugging features are available to help you find the most difficult systems problems.

## Benefits

- **Works with CodeView** --To get you up and going as fast as possible, Soft-ICE is designed to work with your existing software debuggers such as CodeView and Periscope I & II.
- **Breaks the 640K barrier** -- If you have extended memory, Soft-ICE takes up **ZERO** bytes of memory in the first 1MB of address space. This means you can load and debug your largest programs.
- **Power of an in-circuit emulator** -- At only $386, you can give every member of your software development team the power of an in-circuit emulator.
- **AT compatible 80386 PC's** -- Soft-ICE works with all AT compatible 80386 PC's, such as COMPAQ's Deskpro 386 and the IBM Model 80.
- **Easy to learn** -- Soft-ICE is so easy to learn that you can be finding bugs with Soft-ICE by the time you could install a hardware-assisted debugger.

"Since Soft-ICE doesn't take up any of my memory I have it in my AUTOEXEC.BAT to load every day... It has saved me at least one month's time on my latest device driver program." Peter Ricker, President of Maverick Software

30 day money-back satisfaction guarantee. Visa and Master Card accepted. Ask about our coupon program.

**To order or to get more information, call (603) 888-2386**

**NU-MEGA TECHNOLOGIES**
P.O. BOX 7607
NASHUA, NH 03060-7607

CIRCLE NO. 118 ON READER SERVICE CARD

*(continued from page 35)*

ing will be stopped; the reverse is also true.

5. The Control submenu contains several commands that control the operation of the SPA program. Data acquisition can be stopped and started; the display can be cleared; and finally, the SPA program can be reset to its power-on defaults.

6. The Quit menu option returns you to the main SPA display screen, and the End option ends the operation of the SPA program completely and returns control to DOS. Note: Selecting Quit from any submenu will always bring you back up one menu level.

The complete hierarchical menu structure is too big to list. You can look at the procedure *Init_Menu* in the file MENU.PAS for more information. Also, a short textual message is displayed with each possible menu selection. This is a limited form of context-sensitive help to guide you through the menuing system without your having to consult any documentation.

### *The SPA Program's Architecture*

Now that you understand how to use the SPA program, I'll spend a few minutes discussing the underlying technical aspects of the program's operation. For the following discussion, please refer to Figure 4.

As shown in this block diagram, the SPA program is made up of seven relatively independent tasks, most of which are bound to a FIFO. In summary, the tasks perform the following:

The tasks *MoveCOM1Data* and *MoveCOM2Data* perform the same function on different FIFOs for different COM ports. These tasks perform three basic functions. First, they move serial data and handshake information from their respective input FIFO to the opposite output FIFO. This makes the serial data path through the PC. Second, the serial data is tagged with a source identifier (either from COM1 or COM2), and if the display data flag is set, the data is moved into the display FIFO. Finally, code in these tasks provides the data triggering function.

The tasks *OutputCOM1Data* and *OutputCOM2Data* again are identical in function but access different FIFOs. They check to see whether there is data in their respective output FIFOs, and if so they send it out the COM port.

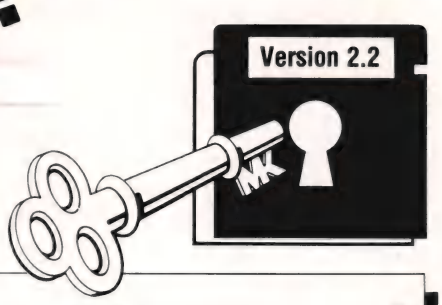
The task *DisplayCOMData* removes entries from the display FIFO, formats them according to the SPA user's specification, and displays them on the SPA's display.

| Bit Numbers | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CD | RI | DSR | CTS | BRK | FE | PE | OR |

***Table 1:*** *Correspondence between data bit encoded and displayed hex byte*

# MASTER*KEY

# Unlocks Everything!




**turn this into this!**

```
C:>DEBUG PROGRAM.COM
-D100 136
8848:0100  EB 18 49 6E 63 6F 72 72-65 63 74 20 44 4F 53 20  k.Incorrect DOS
8848:0110  76 65 72 73 69 6F 6E 0D-0A 24 50 B4 30 CD 21 86  version..$P40M!.
8848:0120  E0 3D 36 01 72 05 3D 0A-02 76 09 BA 02 01 B4 09  `=6.r.=..v.:..4.
8848:0130  CD 21 CD 20 58 EB 2F                             M!M Xk/
-Q
```

```
H00100: JMP     Short H0011A                        ;00100 EB18          __
;-----------------------------------------------
        DB      "Incorrect DOS version"             ;00102 496E636F727265
        DB      0Dh                                 ;00117
        DB      0Ah                                 ;00118
        DB      "$"                                 ;00119 24            $
;-----------------------------------------------
H0011A: PUSH    AX                                  ;0011A 50            P
        MOV     AH,30h                              ;0011B B430          _0
        INT     21h             ;1-DOS_Ver_Number   ;0011D CD21          _!
        XCHG    AH,AL                               ;0011F 86E0          __
        CMP     AX,0136h                            ;00121 3D3601        =6_
        JB      H0012B          ; . . . . . .       ;00124 7205          r_
        CMP     AX,020Ah                            ;00126 3D0A02        =__
        JBE     H00134          ; . . . . . .       ;00129 7609          v_
H0012B: MOV     DX,0102h                            ;0012B BA0201        ___
        MOV     AH,09h                              ;0012E B409          __
        INT     21h             ;1-Display_String   ;00130 CD21          _!
        INT     20h             ;TERM_normally:20h  ;00132 CD20          _
;-----------------------------------------------
H00134: POP     AX                                  ;00134 58            X
        JMP     Short H00166                        ;00135 EB2F          _/
;-----------------------------------------------

MASTER*KEY XREF - PROGRAM.XRF                                    Page 1

0102h                   :   121   2F5   301   320
020Ah                   :   126
03CBh                   :   12B
1-Display_String        :   130   591   610
1-DOS_Ver_Number        :   11D
H00100                  :   100
H0011A                  :   100   11A
H0012B                  :   124   12B
H00134                  :   129   134
H00166                  :   135
TERM_normally:20h       :   132
```

**NOTE: The cross-reference is by memory location within the program file!**

**NOTE: The output is totally Microsoft MASM-compatible.**

(not copy protected)

## MASTER*KEY
## No Other Product Comes Close!

An EXPERT may not know the solution, but always knows where to find it.

MASTER*KEY HELPS ANYONE solve those confusing and frustrating software puzzles more rapidly and easily than any other software available, at any cost! It gives you know-how within hours that may otherwise take years of experience. Create a new program from an old one. DON'T REINVENT THE WHEEL!

**MASTER*KEY - Smart!**
MASTER*KEY is an intelligent self-documenting MS-DOS reverse assembler. Its sophisticated procedures swiftly race through massive and baffling object code files to effortlessly discover potential trouble spots.

**MASTER*KEY - Educational!**
YOU DON'T NEED TO KNOW ASSEMBLY LANGUAGE! MASTER*KEY will take any program from your IBM-compatible computer and return fully-documented, easily-understood assembly language source code (Microsoft MASM 4.0 compatible).

**MASTER*KEY - Easy To Use!**
MASTER*KEY works both automatically from the DOS command line or interactively from menus similar to Lotus Corporation's 1-2-3 or Symphony. No need to remember any new commands or continually refer to a manual. Use it immediately!

**Minimum System Requirements:**
256K + 8088/8086/80186/80286/80386 PC
MS-DOS or PC-DOS 2.0 +
One 360K DSDD Floppy Drive (IBM PC Format)

MS-DOS is a trademark of Microsoft.
PC-DOS is a trademark of IBM.

## MASTER*KEY will guide you step by step to:

1. Help you learn assembly language, if you desire.
2. Discover how any program runs or why it doesn't.
3. Alter or remove unwanted object code from any program.
4. Incorporate routines from compiled programs into other assembly language, Basic, C, or Pascal programs.
5. Make software more compatible with your computer. Be certain a questionable program won't damage your system BEFORE you run it.
6. Modify software to operate with other versions of DOS.
7. Customize your COMMAND.COM or other executable program directly or by reassembling your altered MASTER*KEY source code.

**Order Now! Just $79.95**

Phone orders accepted on MC or VISA
**$82.45** (includes $2.50 shipping)
**$87.65** in California (includes tax & shipping) **C.O.D.** orders add $2.00

**(714) 596-0070**

Please send **MASTER*KEY!**

Send checks to:
**Sharpe Systems Corporation**
2320 E Street, Dept. 44, La Verne, CA 91750

Name _____

Address _____

City _____ State _____ Zip _____

Dealer/Distributor Inquiries Welcome

**Sharpe Systems Corporation**
2320 E Street, Dept. 44, La Verne, CA 91750 714-596-0070

**MASTER*KEY** should not be confused with any public domain or share ware software that may have a similar name or be a similar product.

CIRCLE NO. 119 ON READER SERVICE CARD

PROTOCOL ANALYZER
*(continued from page 37)*

The *Timer* task runs every half second. Its purpose is to update the information on the main SPA display screen. Specifically, it updates the handshake information and the buffer usage information displayed to the user.

The final task, *ProcessKeysTask*, monitors the SPA's keyboard and processes all user keystrokes. The menuing system is invoked via this task.

### The Menuing System

As previously mentioned, a hierarchical series of menus is used to control the operation of the SPA program. The menuing system is modeled after Lotus 1-2-3 because of its ease of use and the appropriateness of its operation. A menu item or submenu is selected by using the cursor arrow keys to place the reverse-video selector box over the desired item and pressing Enter or by typing the first character of the item's name. The user of the SPA program can move through almost the entire menuing sustem by pressing only single keys. In only one instance—the selection of a trigger byte—is the use of more than one key necessary.

Five procedures from the file MENU.PAS (*DisplayMenu*, *ProcessCr*, *ProcessMenu*, *ProcessCmd*, and *DoMenu*) are used in conjunction with a large data structure to provide the Lotus 1-2-3-like menuing system. These procedures implement an A-tree walk through the menu data structure. For more information on the A (or Awkward) tree structure, see the article "Indexing Open Ended Tree Structures" by John Snyder in the May 1984 issue of *Byte* magazine.

The A-tree menu data structure is rather inefficiently, but simply, realized in the SPA program using a three-dimensional array of menu entry records. A menu entry record is used for each item in the entire menu. Figure 6, page 34, shows a portion of the menu structure. Approximately 24K of data is required to support the menuing system as the array of menu entry records is very sparse.

The numbers in ( ) are the array indices of the menu entry record corresponding to this menu item or submenu. All items with an assigned ccode are leaf or terminal nodes of the A tree. These codes are processed by the procedure *ProcessCdm*. Transition characters are those which cause movement to a different menu level. Typically, they are the first character of each menu item. The complete A tree menu structure for the SPA program is built by the procedure Init_Menu. Figure 7, page 34, shows a detailed breakdown of the menu entry record used in the A-tree structure.

The following keys have special significance while using the menus. They are:

Enter—Selects the currently highlighted menu item. If the item is a submenu, the submenu is displayed with its own selectable options. If the item is a leaf node of the tree, it returns a unique command code (*ccode*) specifying an action to be performed.

Cursor arrow keys—Move the highlighted, reverse-video selector box through the items in a menu. Full selection wrapping is supported.

Esc—Terminates the menuing sys-

*(continued from page 39)*

tem and returns to the main SPA display screen.
Home—Moves the selector to the first menu entry.

End—Moves the selector to the last menu entry.

Hopefully, the operation of the menuing system will be discernable from the commented listings. Possibly a future article could be written to discuss the menus in detail if interest warrants it.

### Changes to the Multitasking Kernel

I've made two changes to the kernel presented in the July issue of *DDJ* to augment its capabilities for use with the SPA program. They are:

1. The constant *task_stack_size* has been increased to 1,000 bytes. This was necessary because of procedure nesting depths, the use of interrupts, and Turbo Pascal's use of BIOS calls. Each task is assigned a stack size of 1,000 when forked. The seven tasks utilized for the SPA program consume approximately 7K of memory for their stacks.
2. Substantial portions of the procedures *Yield* and *Wait* have been recoded into assembly language. This minimizes the task-switching overhead experienced by the CPU. Listing Four, page 80, shows the assembly-language recoding. This change was not absolutely necessary for the operation of the SPA, but it seemed to help the performance of the data display when operating above 2,400 baud. By way of comparison, the hand-optimized assembly-language code uses half the instructions generated by the Turbo Pascal compiler for the equivalent code.

### Possible Enhancements

As presented, the SPA program is a rather basic tool. Its present form is in part attributable to the application for which I have been using it. Other applications will require different incarnations of the basic program. Additional features and functions will begin to suggest themselves with prolonged use of the SPA. Already my wish list is growing. Example excerpts are:

1. A setup file that would be read at the start-up of the SPA program that would configure the SPA as I had previously left it. Included in this setup could be serial parameters, trigger info, and display-formatting information.
2. Logging of captured data to a file or printer. Currently, the Shift-PrtScn key sequence must be used to get hard copy of the displayed data.
3. The output of canned serial data to a COM port on the occurrence of a trigger.
4. Increasing the trigger capability from a single byte to multibyte patterns or strings.
5. Other serial data formats—for example, BiSync, HDLC, and SDLC.

Maybe inspired readers can add these functions (and more) and graciously provide me with the code modifications. If I receive enough suggestions and modifications, I'll collect them into a usable form and write another article describing them.

### Modification and Testing Issues

At the present time, the SPA program has never been tested (for any length of time) above 2,400 baud because I lack the equipment to test it throughly. It has been tested substantially at 1,200 baud in the development of an Xmodem protocol. If you require baud rates above 2,400 for your application, you may need to make minor changes to the program to allow it to keep up. With rates above 4,800 baud, I recommend using an IBM PC AT. In the future I hope to test it out thoroughly above 2,400 baud (read, as soon as my project at work slows down some).

As mentioned previously, the software as presented is optimized for the pass through mode of connection. If you would rather use the passive monitoring connection, you can modify the software to improve its performance. The modifications are as follows:

1. Remove the section of code in SPA.PAS (in main) that forks off the *OutputCOM1Data* and *OutputCOM2-*

# Only WATERLOO C's programming environment delivers all 10 of the most wanted user features:

- ☑ High-speed compilation
- ☑ High-performance code
- ☑ Reentrant code options
- ☑ OS linkage options
- ☑ In-line functions
- ☑ ANSI compiler and library
- ☑ Full-screen library
- ☑ Full-screen symbolic debugger
- ☑ Extensive on-line documentation
- ☑ Development tools

**WATERLOO C Version 3.0** is the high-performance development system for IBM 370 mainframe architectures under VM/CMS consisting of:

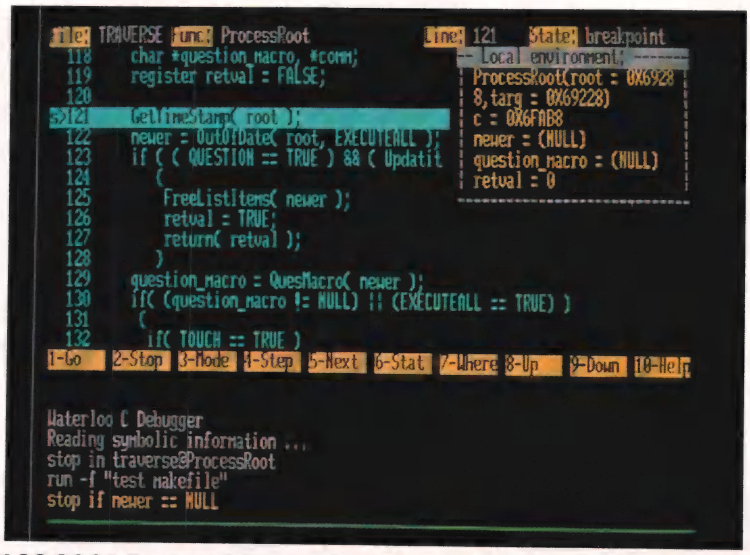
- **Optimizing C Compiler**
- **C Run-time Library**
- **Interactive Source-level Debugger**
- **Development Tools**

**WATERLOO C** is ideal for new software development and for porting existing software to IBM mainframes. It provides ANSI standard support in an integrated CMS environment for portability and efficiency. And it puts all the resources of a full development system at your fingertips so you can quickly achieve quality results.

**WATERLOO C** improves your response time and conserves your mainframe resources with an extremely fast compilation rate. A single phase from source to object means less overhead, while both the compiler and library can reside in shared segments for even faster compiles and links.

**WATERLOO C** generates the highest quality code with an exclusive state-of-the-art optimizer. Options for reentrant applications and OS linkage conventions mean flexibility in development. Moreover, a convenient option allows you to see the original source lines as comments in the assembler output.

The source-level debugger lets you work with your programs the way you wrote them. The full-screen interface feels so natural that it is easy to use. Debugging activity is performed using source variables, functions, and lines. And the execution environment allows stepping, breakpoints, tracing, and display and alteration of your program data.

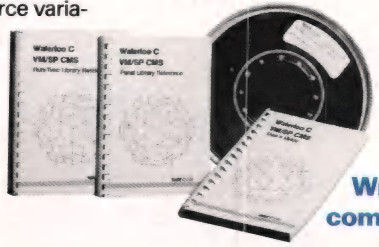



**WATERLOO C 3.0 full-screen debugger: A quick route to quality results.**

Completing the development environment are these important productivity tools:

- Update your programs with a single command: **MAKE.**
- Simplify multi-file maintenance with **GREP** and **DIFF**.
- Track source variables and functions with cross-reference utilities.
- Spot performance bottlenecks with a program execution profiler.

A full support framework provides complete technical backup. Access documentation easily on-line or in comprehensive reference manuals. Our Customer Support Center provides you with expert technical assistance. Newsletters keep you informed about product developments. And on-site installation and training workshops are available.



**Discover the high-performance advantage of WATERLOO C. Write or call for our complete information package:**

**WATCOM**

Find out how quickly and easily you can generate the highest quality code for IBM mainframes by writing or calling for the WATERLOO C complete product information package.

☐ Please send your product information package.

☐ I would like to arrange for an evaluation copy.

Name: _____

Title: _____

Company: _____

Street: _____

City: _____

State: _____ Zip: _____

Telephone: _____

**WATCOM**

Dept. DD-02A
415 Phillip Street
Waterloo, Ontario, Canada N2L 3X2
**Tel. (519) 886-3700**



CIRCLE NO. 122 ON READER SERVICE CARD

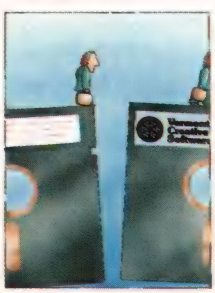
New Prices

OS/2

# WINDOWS FOR DATA®


**If you program in C, take a few moments to learn how Windows for Data can help you build a state-of-the-art user interface.**

- ☑ **C**reate and manage menus, data-entry forms, context-sensitive help, and text displays — all within windows.
- ☑ **D**evelop window-based OS/2 programs right now, without the headaches of learning OS/2 screen management. Run the same source code in PCDOS and OS/2 protected mode.
- ☑ **B**uild a better front end for any DBMS that has a C-language interface (most popular ones do).



## FROM END TO BEGINNING

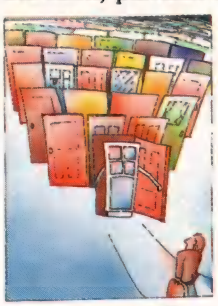
**Windows for Data** begins where other screen packages end, with special features like nested pop-up forms and menus, field entry from lists of choices, scrollable regions for the entry of variable numbers of line items, and an exclusive built-in debugging system.

## NO WALLS

If you've been frustrated by the limitations of other screen utilities, don't be discouraged. You won't run into walls with Windows for Data. Our customers repeatedly tell us how they've used our system in ways we never imagined — but which we anticipated by designing Windows for Data for unprecedented adapatability. You will be amazed at what you can do with Windows for Data.

## YOU ARE ALWAYS IN CHARGE

Control functions that you write and attach to fields and/or keys can read, compare, validate, and change the data values in all fields of the form. Upon entry or exit from any field, control functions can call up subsidiary forms and menus, change the active field, exit or abort the form, perform almost any task you can imagine.



## OUR WINDOWS WILL OPEN DOORS

Our windows will open doors to new markets for your software. High-performance, source-code-compatible versions of Windows for Data are now available for **PCDOS, OS/2, XENIX, UNIX,** and **VMS.** PCDOS versions are fully compatible with Microsoft Windows. No royalties.

You owe it to yourself and your programs to try Windows for Data. If not satisfied, return for a full refund.

Prices: PCDOS $295, Source $295. OS/2 $595. XENIX $795.

**Call: (802) 848-7731** **ext. 31**
Telex: 510-601-4160 VCSOFT FAX 802-848-3502

**Vermont Creative Software**

21 Elm Ave.
Richford,
VT 05476

CIRCLE NO. 123 ON READER SERVICE CARD

*(continued from pagen 40)*

*Data* tasks. These aren't needed for the passive monitor conection because the SPA's PC doesn't have to output any data.

2. Modify the code in the tasks *MoveCOM1Data* and *MoveCOM2Data* so that they don't move the serial data to the output FIFOs. From *MoveCOM1Data*, remove the line:

*PutSerialData(Sd,COM2_Output_Fifo);*

and from *MoveCOM2Data*, remove the line:

*PutSerialData(Sd,COM1_Output_Fifo);*

These changes will eliminate two unnecessary tasks from being executed and will speed up (slightly) the processing of data in two other tasks. The original code is required, however, when the SPA is configured for the pass through method of connection.

### *Availability*

All the source code for articles in this issue is available on a single disk. To order, send $14.95 *to Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

**DDJ**

**(Listings begin on page 44.)**

Vote for your favorite feature/article.
Circle Reader Service **No. 5.**

**Listing One** *(Text begins on page 30.)*

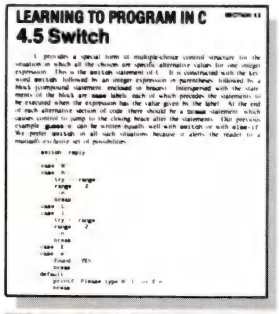
```
{***********************************************}
{***                                         ***}
{***              Turbo Pascal               ***}
{***         Serial Protocol Analyzer        ***}
{***               written by                ***}
{***            Craig A. Lindley             ***}
{***                                         ***}
{***    Ver: 2.0     Last update: 08/15/87    ***}
{***                                         ***}
{***********************************************}

{$K-,U-,C-,G30,D-}

{ ------- Notes on compiler directives --------- }
{ K-     No stack checking otherwise multitasking}
{         kernal will not run.                   }
{ U-,C-  Turn off user break checks to speed     }
{         screen I/O.                            }
{ G30,D- Buffer standard input device (keyboard) }
{         and disable device checks. This makes  }
{         keyboard respond much faster and be    }
{         buffered.                              }

CONST

 HexDigits:  STRING[16] = '0123456789ABCDEF';

 AsciiStrs: ARRAY[0..31] OF STRING[3] =
 ('Nul','Soh','Stx','Etx','Eot','Enq','Ack','Bel',
  'Bs' ,'Ht' ,'Lf' ,'Vt' ,'Ff' ,'Cr' ,'So' ,'Si' ,
  'Dle','Dc1','Dc2','Dc3','Dc4','Nak','Syn','Etb',
  'Can','Em' ,'Sub','Esc','Fs' ,'Gs' ,'Rs' ,'Vs');


 SerialDataFifoSize = 2000;  {serial data fifo size}
 DisplayFifoSize    = 3000;  {display fifo size}

 {$I multi.pas}              {include the}
                             {multitasking kernel}

TYPE

 FullString  = STRING[255];
 Str80       = STRING[80];
 Str3        = STRING[3];

 DataRec = RECORD
  Data,
  Status:  Byte;
 END;

 DisplayRec = RECORD
  Tag: (FromCOM1,FromCOM2);
  DR: DataRec;
 END;

{This fifo overhead structure is the same for}
{all fifo types regardless of the items to be}
{stored in the fifo. Two types are fifos are }
{defined.}

 OverHead = RECORD      {fifo overhead data}
                        {structure}
    Count,              {# of items in fifo}
    Inptr,              {ptr to where items are}
                        {stored}
    Outptr:   Integer;  {ptr to where items are}
                        {fetched}
    NotEmpty,           {ptrs to waiting tasks}
    NotFull:  tcbptr;
 END;


 {definition of a serial data fifo}
 SerialDataFifo = RECORD
   Ovd:  OverHead;             {fifo overhead}
   Data: ARRAY[1..SerialDataFifoSize]
           OF DataRec;         {fifo data area}
 END;

 {definition of display fifo}
 DisplayFifoType = RECORD
   Ovd:  OverHead;
   Data: ARRAY[1..DisplayFifoSize]
           OF DisplayRec;
 END;

 DisplayTriggerType = (Before,After);

VAR

 regs:        register_type;

 {storage for the original IRQ3 & 4 code segment}
 {and instruction pointer addresses}

 OldIRQ3_CS,
 OldIRQ3_IP,
 OldIRQ4_CS,
 OldIRQ4_IP:  Integer;

 {UART status storage variables}

 OldCOM1_Status,
 OldCOM2_Status,
 COM1_Status,
 COM2_Status: Byte;

 {Display formatting boolean flags}

 UpdateScreenStatus,
 AsciiDisplay,
 HandShakeDisplay,
 AddSpace,

 {Data display boolean flags}
 {If true the data from the specified COM port}
 {is tagged and then moved into the display fifo}

 COM1_Display_Data,
 COM2_Display_Data,

 {Data Acquisition boolean flags}
 {Controls acquisiton of data by the Interrupt}
 {Serivce Routines. If true then serial data is}
 {stored by the ISR.}

 COM1_Data_Acquire,
 COM2_Data_Acquire,

 {Variables used for the triggering function}

 TriggerEnabled,
 COM1_Is_Triggered,
 COM2_Is_Triggered:  Boolean;
 TriggerPattern:     Integer;
 TriggerMode:        DisplayTriggerType;

 {Fifo declarations}

 COM1_Input_Fifo,
 COM2_Input_Fifo,
 COM1_Output_Fifo,
 COM2_Output_Fifo:   SerialDataFifo;

 DisplayFifo:        DisplayFifoType;

 {Screen formatting strings built at run time to}
 {format the screen.}

 Line1, Line2,
 Line3, Line4,
 Line5, Line6,
 Line24:        Str80;

 {Cursor storage for DisplayCOMData procedure}

 OldXPos,
 OldYPos:  Integer;

 {Lock for screen control}
 ScreenAccess: Semaphore;

{************ Begin FIFO Procedures ************}

PROCEDURE Init_Fifos;

 PROCEDURE Initialize_fifo(VAR o:OverHead);

 {Initialize a fifo's overhead data structure.}
 {This procedure will work with any type fifo.}
 {This makes the fifo appear empty.}

 BEGIN

    o.Count  := 0;       {count is empty}
    o.Inptr  := 1;       {ptrs to 1st entry}
    o.Outptr := 1;       {put in and take out at}
                         {entry 1}
    o.NotEmpty :=NIL;    {signals to nil}
    o.NotFull  :=NIL;
```

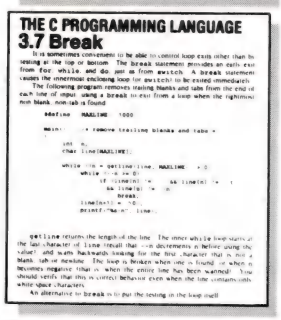
*(continued on page 46)*

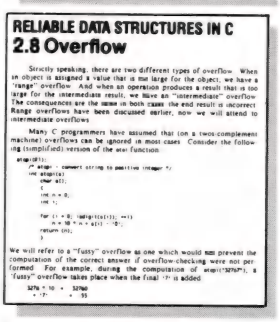
# The "C" Programmer's Set —only $4.95.

## when you join the Library of Computer and Information Sciences.

You simply agree to buy 3 more books—at handsome discounts—within the next 12 months.


LEARNING TO PROGRAM IN C
4.5 Switch


THE C PROGRAMMING LANGUAGE
3.7 Break


RELIABLE DATA STRUCTURES IN C
2.8 Overflow

### Learning to Program in C

*By Thomas Plum.* Beginning programmers will quickly learn how to write C programs that are both efficient and portable. More than an instructive manual, the book also gives insight into the underlying processes of the language and highlights its unique aspects. **Publisher's Price: $25.00.**

### The C Programming Language

*By Brian W. Kernighan and Dennis M. Ritchie.* This best-selling tutorial effectively teaches C through practical, hands-on exercises. Its accessible, step-by-step format enables even novice programmers to take advantage of this powerful, highly-efficient language. A handy C reference manual rounds out this information-packed handbook.

**Publisher's Price: $26.67.**

### Reliable Data Structures in C

*By Thomas Plum.* This example-filled manual details all the advanced features of C. Coverage includes practical case studies illustrating sophisticated constructs, general-purpose functions and macros, and application-oriented case studies. A series of reliability rules help you write programs that deliver "no surprises." Concepts from the ANSI standardization of C point the way to powerful but portable applications. **Publisher's Price: $25.00.**


The Library of Computer and Information Sciences is the oldest, largest book club especially designed for computer professionals. In the incredibly fast-moving world of data processing, where up-to-the-moment knowledge is essential, we make it easy to keep totally informed on all areas of the information sciences. What's more, our selections offer you discounts of up to 30% or more off publishers' prices.

## 4 Good Reasons to Join

**1. The Finest Books.** Of the hundreds of books submitted to us each year, only the very finest are selected and offered. Moreover, our books are always of equal quality to publishers' editions, *never* economy editions.

**2. Big Savings.** In addition to getting the "C" Programmer's Set for only $4.95 when you join, you keep saving substantially, up to 30% and occasionally even more. (For example, your total savings as a trial member—including this introductory offer—can easily be over 50%. That's like getting every other book free!)

**3. Bonus Books.** Also, you will immediately become eligible to participate in our Bonus Book Plan, with savings of 65% off the publishers' prices.

**4. Convenient Service.** At 3–4 week intervals (16 times per year), you will receive the Library of Computer and Information Sciences News, describing the Main Selection and Alternate Selections, together with a dated reply card. If you want the Main Selection, do nothing, and it will be sent to you automatically. If you prefer another selection, or no book at all, simply indicate your choice on the card and return it by the date specified. You will have at least 10 days to decide. If, because of late mail delivery of the News, you should receive a book you do not want, we guarantee return postage.

**The Library of Computer and Information Sciences** 7-EV2
Riverside, N.J. 08075

Please accept my application for trial membership and send me the "C" Programmer's Set (00772) billing me only $4.95, plus shipping and handling. I agree to purchase at least three additional Selections or Alternates over the next 12 months. Savings range up to 30% and occasionally even more. My membership is cancelable any time after I buy these three additional books. A shipping and handling charge is added to all shipments.

**No-Risk Guarantee:** If I am not satisfied—for any reason—I may return the "C" Programmer's Set within 10 days. My membership will be canceled, and I will owe nothing.

Name_____

Name of Firm_____
(If you want subscription sent to your office)

Address_____ Apt.____

City_____

State_____ Zip_____

(Books purchased for professional purposes may be a tax-deductible expense. Offer good in Continental U.S. and Canada only. Prices slightly higher in Canada.)

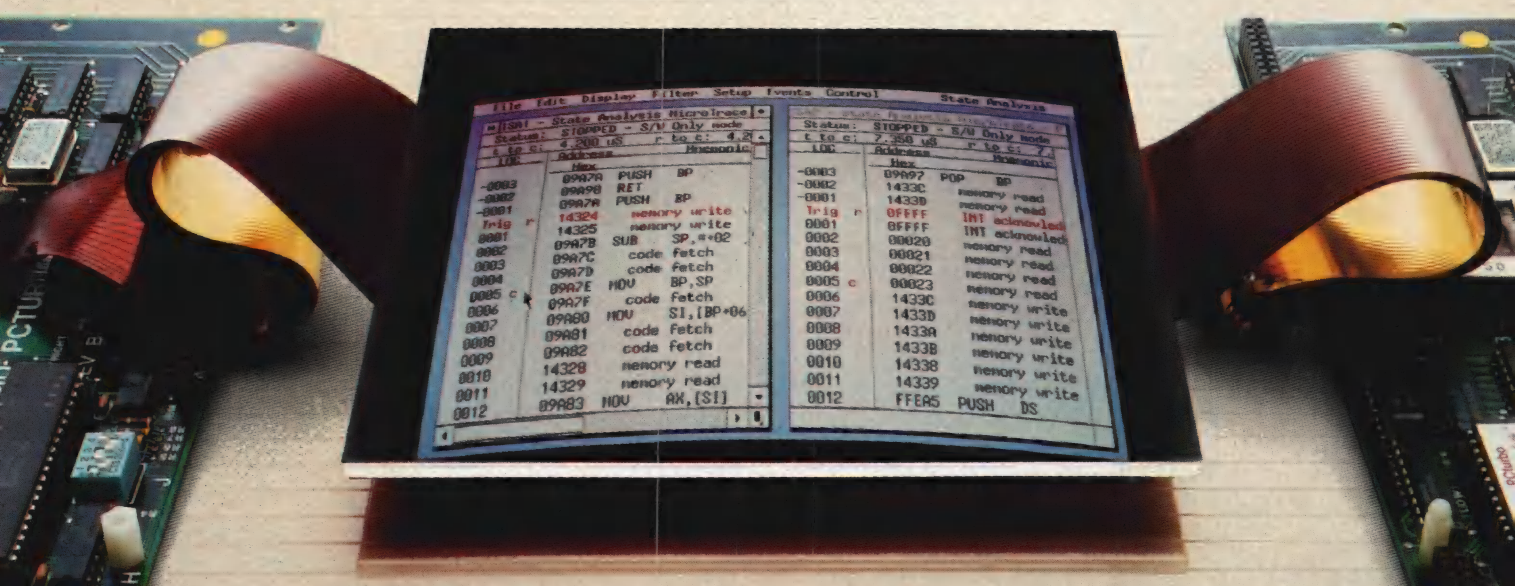
Dr. Dobb's Journal 2/88

**Listing One** *(Listing continued, text begins on page 30.)*

```
 END;

BEGIN

   Initialize_fifo(COM1_Input_fifo.Ovd);
   Initialize_fifo(COM1_Output_fifo.Ovd);
   Initialize_fifo(COM2_Input_fifo.Ovd);
   Initialize_fifo(COM2_Output_fifo.Ovd);
   Initialize_fifo(DisplayFifo.Ovd);

END;


PROCEDURE PutSerialData (d:DataRec;
                         VAR f:SerialDataFifo);

BEGIN

   WITH f.Ovd DO
   BEGIN                {check if fifo full}
      IF Count = SerialDataFifoSize THEN
      BEGIN             {if so go to sleep}
         waitfor := addr (NotFull);
         wait;
      END;              {when not full add}
      Count:=Count+1;   {one more to count}
      f.data[Inptr]:=d; {store the data record}
      Inptr:=Inptr+1;   {bump input pointer}
      IF Inptr > SerialDataFifoSize THEN
         Inptr:=1;      {wrap ptr if necessary}

      {if waiters for this fifo wake them}

      IF NotEmpty <> NIL THEN
         send(NotEmpty);
   END;

END;


PROCEDURE GetSerialData (VAR f:SerialDataFifo;
                         VAR d:DataRec);

BEGIN

   WITH f.Ovd DO
   BEGIN                {check if fifo empty}
      IF Count = 0 THEN
      BEGIN             {if so go to sleep}
         waitfor := addr (NotEmpty);
         wait;
      END;
                        {when data is available}
      Count:=Count-1;   {one less to count}
      d :=f.data[Outptr];  {get the data record}
      Outptr:=Outptr+1;{bump output pointer}
      IF Outptr > SerialDataFifoSize THEN
         Outptr:=1;     {wrap ptr if necessary}
      {if waiters for this fifo wake them}

      IF NotFull <> NIL THEN
         send(NotFull);
   END;

END;


PROCEDURE PutDisplayData (d:DisplayRec;
                          VAR f:DisplayFifoType);

BEGIN

   WITH f.Ovd DO
   BEGIN                {check if fifo full}
      IF Count = DisplayFifoSize THEN
      BEGIN             {if so go to sleep}
         waitfor := addr (NotFull);
         wait;
      END;              {when not full add}
      Count:=Count+1;   {one more to count}
      f.data[Inptr]:=d; {store the data record}
      Inptr:=Inptr+1;   {bump input pointer}
      IF Inptr > DisplayFifoSize THEN
         Inptr:=1;      {wrap ptr if necessary}

      {if waiters for this fifo wake them}

      IF NotEmpty <> NIL THEN
         send(NotEmpty);
   END;

END;


PROCEDURE GetDisplayData (VAR f:DisplayFifoType;
                          VAR d:DisplayRec);

BEGIN

   WITH f.Ovd DO
   BEGIN                {check if fifo empty}
      IF Count = 0 THEN
      BEGIN             {if so go to sleep}
         waitfor := addr (NotEmpty);
         wait;
      END;
                        {when data is available}
      Count:=Count-1;   {one less to count}
      d :=f.data[Outptr];  {get the data record}
      Outptr:=Outptr+1;{bump output pointer}
      IF Outptr > DisplayFifoSize THEN
         Outptr:=1;     {wrap ptr if necessary}

      {if waiters for this fifo wake them}

      IF NotFull <> NIL THEN
         send(NotFull);

   END;

END;

{Include the menuing system}

{$I menu.pas}

{Include the serial procedures}

{$I serial.pas}

{********* Additional Serial Procedures *********}


PROCEDURE SetBreak (PortAddr:Integer; State:Boolean);

{Controls the break generation for the specified}
{COM port.}

VAR

 Temp:  Byte;

BEGIN

   {Read the LineControl reg of the 8250}
   {either set or reset the break bit D6}
   {as specified. Write the new reg value}
   {back to the port}

   Temp := port[PortAddr+LineControl];
   IF State THEN
      Temp := Temp OR $40
   ELSE
      Temp := Temp AND $BF;
   port[PortAddr+LineControl] := Temp;

END;


PROCEDURE MakeHandShake (PortAddr:Integer;
                         Status:Byte);

VAR

 Temp:   Byte;

BEGIN

   {Set the bits in the ModemControl reg of the}
   {specified COM port according to the bits}
   {of the variable Status. This procedure is}
   {used to always force the handshake lines of}
   {the COM ports to agree}

   Temp := port[PortAddr+ModemControl];

   IF (Status AND BrkBit) <> 0 THEN
      SetBreak(PortAddr,True)
   ELSE
      SetBreak(PortAddr,False);
```

*(continued on page 48)*

# At last, multi-processor time-aligned trace.



Since so many embedded microcomputer designs employ more than one processor, you'd think there would be many tools to trace multi-processor dialog in both assembly-level and high-level code. And then time-align the results.

In fact, there's only one. The Software Analysis Workstation™ (SAW) from NWIS.

The SAW alone gives you a choice of a time-aligned, assembly-level trace on both processors. Or a time-aligned trace of module/procedure events on one processor and assembly-level activity on a second processor.

At last, you have a fast, effective means for evaluating complex, inter-processor transactions as they occur in real-time in the target system.

The SAW's dual-processor trace is displayed in a fully time-synchronized manner, so you can instantly analyze relationships between the two execution streams. Scroll the trace from one processor, and the trace from the other processor can be locked to scroll right along with it. The timing relationship between the two traces automatically stays intact, even if the two processors have different clock frequencies.

And multi-processor, time-aligned trace is only one of the SAW's outstanding debug, optimization and verification functions. You also get real-time, non-intrusive performance analysis. And CodeMap,™ for rapid and complete verification of code coverage. All these tools are compiler/assembler and host development computer independent.

The SAW is already producing dramatic results for software engineers around the world. Let us connect the SAW to your target system and you'll see why. For an on-site demo and free applications library, phone: 1-800-547-4445

NWIS

NORTHWEST INSTRUMENT SYSTEMS, INC.

P.O. Box 1309 • Beaverton, OR 97075 • 1-800-547-4445

CIRCLE NO. 125 ON READER SERVICE CARD

AUSTRIA: 0222-253626 BELGIUM: 02-2418130 DENMARK: 05-611100 FINLAND: 0-5284316 FRANCE: 014-5347535 or 016-9070935 ISRAEL: 03-403373 ITALY: 02-470646 THE NETHERLANDS: 01720-43221
NORWAY: 02-789460 SPAIN: 01-4558112 SWEDEN: 08-7390045 SWITZERLAND: 01-7231410 UNITED KINGDOM: 06284-4426 NWIS-USA: 503-690-1300 WEST GERMANY: 089-858020

**Listing One** *(Listing continued, text begins on page 30.)*

```
    IF (Status AND CTSBit) <> 0 THEN
       Temp := Temp OR $02
    ELSE
       Temp := Temp AND $FD;

    IF (Status AND DSRBit) <> 0 THEN
       Temp := Temp OR $01
    ELSE
       Temp := Temp AND $FE;

    {Store the new value of the ModemControl}
    {register back}

    port[PortAddr+ModemControl] := Temp;

END;


{************** Display Procedures *************}

PROCEDURE BuildDisplay;

{Setup the main SPA display screen}
{The OldCOM?_Status variables are consciencely}
{clobbered to force the screen to be updated}
{after it is built or rebuilt after the menus}
{are displayed}

BEGIN

   OldCOM1_Status    := $FF;
   OldCOM2_Status    := $FF;

   WriteStringAt(Line1,High,1,1);
   WriteStringAt(Line2,Low,1,2);
   WriteStringAt(Line3,Low,1,3);
   WriteStringAt(Line4,Low,1,4);
   WriteStringAt(Line5,Low,1,5);
   WriteStringAt(Line6,Low,1,6);
   WriteStringAt(Line24,Rev,1,24);

END;
```

```
PROCEDURE DisplayHandShakeStatus (PortAddr:Integer;
                                  InStatus,
                                  OutStatus:Byte);

CONST

 StatLineNum = 3; {screen line of line status}
 InLineNum   = 4; {screen line of in handshake}
                  {lines}
 OutLineNum  = 5; {screen line of out handshake}
                  {lines}

 CDOffset  = 11;  {offsets on a screen line for}
 RIOffset  = 16;  {the individual items to be}
 DSROffset = 22;  {displayed}
 CTSOffset = 28;
 BRKOffset = 13;
 FEOffset  = 18;
 PEOffset  = 23;
 OROffset  = 28;
 DTROffset = 12;
 RTSOffset = 22;

VAR

 DisplayOffset:  Integer;
 Ind:            Char;

BEGIN

   {Where an item is displayed is determined in}
   {part by which COM status is being displayed}
   {COM1's offset is 0 whereas COM2's offset is}
   {51 character positions.}

   IF PortAddr = COM1 THEN
      DisplayOffset := 0
   ELSE
      DisplayOffset := 51;

   IF (InStatus AND BrkBit) <> 0 THEN
      Ind := 'B'
   ELSE
      Ind := '-';
   WriteStringAt(Ind,High,
                 BRKOffset+DisplayOffset,
                 StatLineNum);
```

*(continued on page 51)*

# All the Tools You Need For Motorola 680X0 From Whitesmiths

***Whitesmiths, Ltd. now offers a complete set of 68K Cross Development Tools — specifically designed to work together — for the Motorola 68000 family of microprocessors. You get:***

*A C CROSS COMPILER*
Whitesmiths' C Compilers offer the closest conformance currently available to the draft ANSI C Standard. We've added 68020 and 68881 support, and dramatically optimized code generation, so you can get the code quality you need today with the language you'll need tomorrow.

*SUPPORT TOOLS*
We have all the extras you need to develop embedded programs. Our powerful object utilities help you link multi-segment programs, build direct and sequential libraries, create load maps and interspersed listings, and talk to dozens of downloaders, emulators, and PROM programmers.

*C SOURCE LEVEL DEBUGGING*
We have the support you need to debug in terms of C functions, data types, and source lines. You debug what you write, not a lower level language.

*A MICSIM SIMULATOR*
You can debug your embedded programs right on your development host — our MICSIM Simulator needs no extra hardware. It's like debugging on your favorite emulator, but with no contention for dedicated resources, no download time, and with the symbolic breakpoint and trace control you've always dreamed of having.

*AN XA8 CROSS ASSEMBLER*
Our macro assembler is both fast and powerful, with support for 68020, 68881, and 68551.

*A PASCAL COMPILER*
You can program as much as you want in ISO Standard Pascal, or use the powerful extensions we've added to this production quality compiler. And you get complete integration with C and assembly language as well.

***Working together, the 68K Cross Development Tools deliver both optimized performance and improved programmer productivity. Best of all, Whitesmiths offers everything you need at a very competitive price. We've been delivering and supporting high quality software development tools since 1978, and we're committed to continually enhancing our product line.***

***If you develop 68000 programs on a DEC VAX, an IBM PC, or a UNIX workstation, chances are we can save you time and money. For more technical details, call our toll-free number today. We also offer attractive packages for OEMs.***


**Whitesmiths, Ltd.**
59 Power Road
Westford, MA 01886
617/692-7800

TOLL-FREE NUMBER 1-800-225-1030

**CIRCLE NO. 201 ON READER SERVICE CARD**

# ALL GAIN, NO PAIN

## Blow away the 640K barrier

Gain the benefits of protected mode the easy way with OS/286™ and OS/386™. These tools for C, Fortran, Pascal and Assembly language programmers permit rapid conversion of existing DOS applications from "real" 8086 mode to "protected" 286 and 386 mode. They don't replace or modify DOS, but extend it to protected mode.

OS/286 and OS/386 are the only DOS extenders that span both the 286 and 386 processors, with 32-bit capability *today* on 386s that yields twice the performance of 16-bit mode. OS/286 and OS/386 have quickly become the preferred solution for developers of high performance, memory-intensive applications, including CADKEY, CASE, VIEWlogic, and Gold Hill, and premier language developers Lahey and Metaware.

Our optional TOUCHDOWN™ BIOS supplement provides fast and reliable protected mode operation on *any* 286 system, even those with problems resetting the 286. (Ever notice how few existing machines Operating System/2 runs on?)

If your applications are running out of memory or need more speed, don't wait for the "solution" that means abandoning your investment in DOS. Enhance them now with OS/286 and OS/386 — *products not promises.*

**Make big programs run faster in protected mode**

**OS/286™ & OS/386™ Benefits:**

- Gain multi-megabytes of directly addressable memory (15-Mb-286, 4Gb-386)
- Increase performance by eliminating overlays and EMS
- Convert in days, not months
- Continue to work with a DOS interface and use existing TSRs, device drivers, graphic routines, etc.
- Stay compatible with the widest array of systems

A.I. Architects Software Developers Kit **$495**

includes full support for:

- MetaWare High C (16 & 32 bit)
- Professional Pascal (16 & 32 bit)
- Lahey F77L FORTRAN
- Microsoft C & Fortran 4.0,
- MASM, MS-Link
- Phoenix PLINK86
- Halo & GSS Graphics
- Pharlap 386: ASM/LINK

more to come

Run time licenses for OS/286 and OS/386 are available at nominal cost.

**The HummingBoard™ turns any XT or AT into the fastest 386 system available. The dual processor architecture boosts performance significantly over comparable single processor systems or accelerator boards. Available with 2 to 24Mb RAM, 16 or 20Mhz speed, and 387 floating point coprocessor.**

**A.I. Architects, Inc.**
One Kendall Square, Building 400
Cambridge, Massachusetts 02139
(617) 577-8052

**Listing One** *(Listing continued, text begins on page 30.)*

```
  IF (InStatus AND FEBit) <> 0 THEN
     Ind := 'E'
  ELSE
     Ind := '-';
  WriteStringAt(Ind,High,
                FEOffset+DisplayOffset,
                StatLineNum);

  IF (InStatus AND PEBit) <> 0 THEN
     Ind := 'E'
  ELSE
     Ind := '-';
  WriteStringAt(Ind,High,
                PEOffset+DisplayOffset,
                StatLineNum);

  IF (InStatus AND ORBit) <> 0 THEN
     Ind := 'E'
  ELSE
     Ind := '-';
  WriteStringAt(Ind,High,
                OROffset+DisplayOffset,
                StatLineNum);

  IF (InStatus AND CDBit) <> 0 THEN
     Ind := 'M'
  ELSE
     Ind := 'S';
  WriteStringAt(Ind,High,
                CDOffset+DisplayOffset,
                InLineNum);

  IF (InStatus AND RIBit) <> 0 THEN
     Ind := 'M'
  ELSE
     Ind := 'S';
  WriteStringAt(Ind,High,
                RIOffset+DisplayOffset,
                InLineNum);

  IF (InStatus AND DSRBit) <> 0 THEN
     Ind := 'M'
  ELSE
     Ind := 'S';
  WriteStringAt(Ind,High,
                DSROffset+DisplayOffset,
                InLineNum);

  IF (InStatus AND CTSBit) <> 0  THEN
     Ind := 'M'
  ELSE
     Ind := 'S';
  WriteStringAt(Ind,High,
                CTSOffset+DisplayOffset,
                InLineNum);

  {Note: The OTHER COM port is consulted about}
  {the output status. DTR of this port should}
  {equal DSR of other port. RTS of this port}
  {should equal CTS of other port}

  IF (OutStatus AND DSRBit) <> 0 THEN
     Ind := 'M'
  ELSE
     Ind := 'S';
  WriteStringAt(Ind,High,
                DTROffset+DisplayOffset,
                OutLineNum);

  IF (OutStatus AND CTSBit) <> 0  THEN
     Ind := 'M'
  ELSE
     Ind := 'S';
  WriteStringAt(Ind,High,
                RTSOffset+DisplayOffset,
                OutLineNum);

END;


PROCEDURE DisplayBufferStatus;

VAR

 PerCentage:  Integer;
 PerCentStr:  Str80;

BEGIN

   PerCentage := Round((COM1_Input_Fifo.Ovd.Count/
                        SerialDataFifoSize) * 100);
   Str(PerCentage:2,PerCentStr);
   WriteStringAt(PerCentStr + '%',High,12,2);

   PerCentage := Round((COM1_Output_Fifo.Ovd.Count/
                        SerialDataFifoSize) * 100);
   Str(PerCentage:2,PerCentStr);
   WriteStringAt(PerCentStr + '%',High,26,2);

   PerCentage := Round((COM2_Input_Fifo.Ovd.Count/
                        SerialDataFifoSize) * 100);
   Str(PerCentage:2,PerCentStr);
   WriteStringAt(PerCentStr + '%',High,63,2);

   PerCentage := Round((COM2_Output_Fifo.Ovd.Count/
                        SerialDataFifoSize) * 100);
   Str(PerCentage:2,PerCentStr);
   WriteStringAt(PerCentStr + '%',High,77,2);

   PerCentage := Round((DisplayFifo.Ovd.Count/
                        DisplayFifoSize) * 100);
   Str(PerCentage:2,PerCentStr);
   WriteStringAt(PerCentStr + '%',High,47,5);

END;


FUNCTION FormatCharAscii (Num:Integer) : Str3;

BEGIN

    IF Num < ORD(' ') THEN
        FormatCharAscii := AsciiStrs[Num]
    ELSE
        FormatCharAscii := chr(Num);

END;
```

*(continued on next page)*

**Listing One** *(Listing continued, text begins on page 30.)*

```
FUNCTION FormatCharHex (Num:Integer) : Str3;

BEGIN

   FormatCharHex :=
      HexDigits[(Num SHR 4) AND $000F +1 ] +
      HexDigits[(Num AND $000F) + 1];

END;


FUNCTION DisplayData (D:DisplayRec) : Integer;

VAR

 VideoAttrib:  AttribType;
 Temp:         STRING[7];

BEGIN

   WITH D DO
   BEGIN
      {If data from either channel should be}
      {displayed then...}

      IF (((Tag = FromCOM1) AND COM1_Display_Data)  OR
          ((Tag = FromCOM2) AND COM2_Display_Data)) THEN
      BEGIN
         {set video attribute depending upon}
         {which COM channel it is from}
         IF Tag = FromCOM1 THEN
            VideoAttrib := Low
         ELSE
            VideoAttrib := Rev;

         {choose ASCII or Hex format}
         IF AsciiDisplay THEN
            Temp := FormatCharAscii(DR.Data)
         ELSE
            Temp := FormatCharHex(DR.Data);

         IF HandShakeDisplay THEN
            Temp := Temp + ';$'+
               FormatCharHex(DR.Status);

         {write serial data string to display}
         WriteString(Temp,VideoAttrib);

         {if formatting with spaces}
         IF AddSpace THEN
         BEGIN
            {output space to display and add}
            {a space to string for length calc}
            write(' ');
            Temp := Temp + ' ';
         END;

         {ret length of formatted item}
         DisplayData := length(Temp);
      END
      ELSE
         {if no data ret 0 length}
         DisplayData := 0;
   END;

END;

{*********** Miscellaneous Procedures ***********}

PROCEDURE Init_Program;

{Perform default initialization for protocol}
{analyzer program}

BEGIN

   Init_Fifos;

   COM1_Data_Acquire := True;
   COM2_Data_Acquire := True;

   COM1_Display_Data := True;
   COM2_Display_Data := True;
```

*(continued on page 54)*

# The Ada® world just changed. Again.

Meridian continues its tradition of Ada "firsts" by introducing a powerful new set of integrated software development tools.

**AdaVantage v2.1 Optimizing Ada Compiler** Available for the IBM PC and compatibles and the Apple MacIntosh. An exceptionally fast validated Ada compiler that for the first time brings a true world-class production quality Ada compiler to your desktop personal workstation.

**AdaVantage Debugger** An interactive source-level debugger for use with programs written using the Meridian AdaVantage compiler. The debugger allows the programmer complete control over the execution of an Ada program in high-level Ada terms — no knowledge of the underlying machine architecture is required. The debugger supports breakpoints, subprogram traces, single-stepping, call backtraces, and full Ada reference syntax.

**Ada Developer Interface** A powerful interactive screen-oriented interface to the Ada library dependency information. Includes built-in operations to edit, "pretty print", compile, and link any Ada library unit. Configuration file allows user tailoring of all operations and displays.

**Run-Time Customization Library** For preparation of Ada application programs for execution on 80x86-based embedded systems or other operating systems. The Library is a collection of Ada source files, batch files, and documentation that define the customizable, system-dependent components of the AdaVantage Run-Time System.

**AdaStarter** Identical to the validated v2.1 AdaVantage compiler with limitations that permit up to ten library units, each with up to two-hundred executable statements (unlimited declarations and comments). This is approximately equivalent to a 4000 line program. The price of AdaStarter is applicable towards purchase of the AdaVantage production compiler. Get the full power of Ada for only $99!

**AdaDesigner** A collection of tools supporting the design, programming, and documentation phases of the software life cycle. Tools include a structured editor/synthesizer coupled to a text editor/incremental syntax analyzer for Ada, an incremental editor for design languages, a program derivation processor that guarantees your design is always in sync with your implementation, and a structured documentation generator.

**Configuration** The compilers all run in a standard PC configuration with 640K of memory (1MB on the MacIntosh) and a hard disk.

For more information call 1-800-221-2522.
In California, call 714-380-9800.

> ***"Meridian Software Systems AdaVantage version 2.0 compiler is an Ada software milestone."*** — PC Week

MERIDIAN
SOFTWARE SYSTEMS, INC.

23141 VERDUGO DRIVE • SUITE 105 • LAGUNA HILLS, CALIFORNIA 92653
800/221-2522 (Outside California) • 714/380-9800 (Inside California)
Telex: 650-268-0547 MCI • Fax: 714/380-1683



CIRCLE NO. 129 ON READER SERVICE CARD

# PROTOCOL ANALZYER

**Listing One** *(Listing continued, text begins on page 30).*

```
    TriggerEnabled    := False;
    COM1_Is_Triggered := False;
    COM2_Is_Triggered := False;

    AsciiDisplay      := True;
    HandShakeDisplay  := False;
    AddSpace          := True;

    COM1_Status       := 0;
    COM2_Status       := 0;

    {set serial com defaults 1200 baud, 8 bit word,
    {1 stopbit, no parity}
    {both COM ports always set the same}

    COM_Rate     := 8;          {1200 baud}
    COM_StopBits := 1;
    COM_DataBits := 8;
    COM_Parity   := none;

    SetNewCOMParameter; {apply the parameters}

END;


PROCEDURE VerifyHardware;

CONST

 TestByte = $5A;   {try to store and retrieve}
                   {this value}
 SafeByte = $03;   {set default 8 bits 1 stop}

BEGIN

   ClrScr;
   {just in case of reentry}
   Disable_Serial_Devices;

   WriteString('  Serial Protocol Analyzer Hardware
                Verification Check  ',Rev);
   WriteStringAt('Checking COM1',HighBlink,2,3);
   port[COM1+LineControl] := TestByte;
   delay(700);
   IF port[COM1+LineControl] <> TestByte THEN
   BEGIN
      WriteStringAt('COM1 Hardware Bad or Missing',
                    Rev,2,3);
      Halt;
   END;
   port[COM1+LineControl] := SafeByte;
   WriteStringAt('COM1 Hardware Verified',
                 Low,2,3);

   delay(700);
   WriteStringAt('Checking COM2',HighBlink,
                 2,4);
   port[COM2+LineControl] := TestByte;
   delay(500);
   IF port[COM2+LineControl] <> TestByte THEN
   BEGIN
      WriteStringAt('COM2 Hardware Bad or Missing',
                    Rev,2,4);
      Halt;
   END;
   port[COM2+LineControl] := SafeByte;
   WriteStringAt('COM2 Hardware Verified',Low,2,4);

   delay(2000);
   ClrScr;

END;


{****** Menu Command Processing Procedure *******}

{Declared forward previously}

PROCEDURE ProcessCmd;
```

*(continued on page 56)*

# New Clipper.™ Because your dBase applications keep getting bigger and better.

Get Clipper's newest release, Summer '87, for far faster program execution, smoother, speedier development cycles and faster compiling than ever before.

The more sophisticated your database applications become, the more you need the Clipper compiler.

**More Programming Power.**
New commands, functions and dBASE® compatible indexing make new Clipper the most powerful solution yet for creating the types of dBASE III PLUS™ applications you're working on today. Single-user or networking. For example, fully programmable functions make it easy to browse records, create pop-up menus or edit free text.

It's easy to develop your own user-defined functions and libraries with the Microsoft® C compiler, assembly routines or Clipper itself, and incorporate them into your application. And new Clipper compiles your code five times faster than its predecessor.

Bottom line: You get more sophisticated, better applications, written quicker.

And you *still* don't have to buy extra copies of dBASE for every user of your Clipper-compiled program, or pay runtime or licensing fees.

**Faster Program Execution.**
dBASE can bog down in complex applications. Not with new Clipper. It's 10 to 20 times faster than simple dBASE code — and even twice as fast as last year's Clipper.

**Update today.**
If you're currently a Clipper user, update to new Clipper for as little as $125. Included is a new manual complete with a full dBASE language reference section.

If you're not yet using Clipper, there's never been a better time to step up to the best.

To receive your new Clipper update, or for a **free demo diskette,** call (213) 390-7923 today. That's Clipper. The better you get, the better *we* get.

Nantucket Corporation. 12555 W. Jefferson Blvd., Los Angeles, CA 90066.



Nantucket®

**Listing One** *(Listing continued, text begins on page 30.)*

```
VAR

 HexStr:    Str3;
 HexChar,
 ErrCode:   Integer;
 Ch:        Char;

BEGIN

   IF cmd_code <> 0 THEN
   BEGIN
      CASE cmd_code OF
         1: ExitMenu := True;
         2: BEGIN   {300 baud}
               COM_Rate := 6;
               SetNewCOMParameter;
            END;

         3: BEGIN   {600 baud}
               COM_Rate := 7;
               SetNewCOMParameter;
            END;

         4: BEGIN   {1200 baud}
               COM_Rate := 8;
               SetNewCOMParameter;
            END;

         5: BEGIN   {2400 baud}
               COM_Rate := 11;
               SetNewCOMParameter;
            END;

         6: BEGIN   {4800 baud}
               COM_Rate := 13;
               SetNewCOMParameter;
            END;

         7: BEGIN   {9600 baud}
               COM_Rate := 15;
               SetNewCOMParameter;
            END;

         8: BEGIN   {1 stop bit}
               COM_StopBits := 1;
               SetNewCOMParameter;
            END;

         9: BEGIN   {2 stop bit}
               COM_StopBits := 2;
               SetNewCOMParameter;
            END;

        10: BEGIN   {5 bit word}
               COM_DataBits := 5;
               SetNewCOMParameter;
            END;

        11: BEGIN   {6 bit word}
               COM_DataBits := 6;
               SetNewCOMParameter;
            END;

        12: BEGIN   {7 bit word}
               COM_DataBits := 7;
               SetNewCOMParameter;
            END;

        13: BEGIN   {8 bit word}
               COM_DataBits := 8;

               SetNewCOMParameter;
            END;

        14: BEGIN   {Odd Parity}
               COM_Parity := Odd;
               SetNewCOMParameter;
            END;

        15: BEGIN   {Even Parity}
               COM_Parity := Even;
               SetNewCOMParameter;
            END;

        16: BEGIN   {No Parity}
               COM_Parity := None;
               SetNewCOMParameter;
            END;

        17: BEGIN   {Display COM1 only}
               COM1_Display_Data := True;
               COM2_Display_Data := False;
            END;

        18: BEGIN   {Display COM2 only}
               COM1_Display_Data := False;
               COM2_Display_Data := True;
            END;

        19: BEGIN   {Display both COM1 and COM2}
               COM1_Display_Data := True;
               COM2_Display_Data := True;
            END;

        20: BEGIN   {COM1 is triggered}
               GoToXY(1,25);
               ClrEol;
               COM1_Is_Triggered := True;
               COM2_Is_Triggered := False;
               WriteStringAt('COM1 awaiting trigger',
                             HighBlink,3,25);
            END;

        21: BEGIN   {COM2 is triggered}
               GoToXY(1,25);
               ClrEol;
               COM1_Is_Triggered := False;
               COM2_Is_Triggered := True;
               WriteStringAt('COM2 awaiting trigger',
                             HighBlink,3,25);
            END;

        22: BEGIN   {Input trigger pattern}
               GoToXY(1,MenuLine4);  {position cursor}
               ClrEol;               {clear line}
               WriteString(
                'Input trigger pattern as two hex
                 digits: ',High);

               {initialize str with hex prefix}
               HexStr := '$';
               FOR HexChar := 1 TO 2 DO
               BEGIN
                  REPEAT
                     {get a char}
                     Ch := upcase(Char(GetKey));
                     {loop until valid char is input}
                  UNTIL pos(Ch,HexDigits) <> 0;
                  {display to user}
                  write(Ch);
                  {add to hex string}
                  HexStr := HexStr + Ch;
               END;
               {convert hex to int}
               Val(HexStr,TriggerPattern,ErrCode);
               GoToXY(28,25);
               Write('Trigger Pattern: ',
                     FormatCharHex(TriggerPattern));
            END;

        23: BEGIN   {Display BEFORE Trigger mode}
               TriggerMode := Before;
               WriteStringAt('Mode: Display Before Trigger'
                             ,Low,51,25);
               IF COM1_Is_Triggered THEN
                  {start with displaying data}
                  COM1_Display_Data := True
               ELSE
               IF COM2_Is_Triggered THEN
                  COM2_Display_Data := True
               ELSE
                  WriteStringAt('Mode Error --
                                          Select channel'
                                ,HighBlink,51,25);
            END;

        24: BEGIN   {Display AFTER Trigger mode}
               TriggerMode := After;
               WriteStringAt('Mode: Display After Trigger'
                             ,Low,51,25);
               IF COM1_Is_Triggered THEN
                  {start without displaying data}
                  COM1_Display_Data := False
               ELSE
               IF COM2_Is_Triggered THEN
                  COM2_Display_Data := False
               ELSE
               WriteStringAt('Mode Error -- Select channel'
                             ,HighBlink,51,25);
            END;
```


*(continued on page 59)*

# "How to protect your software by letting people copy it."

*By Dick Erett, President of Software Security*

Inventor and entrepreneur, Dick Erett, explains his company's view on the protection of intellectual property.

"A crucial point that even sophisticated software development companies and the trade press seem to be missing or ignoring is this:

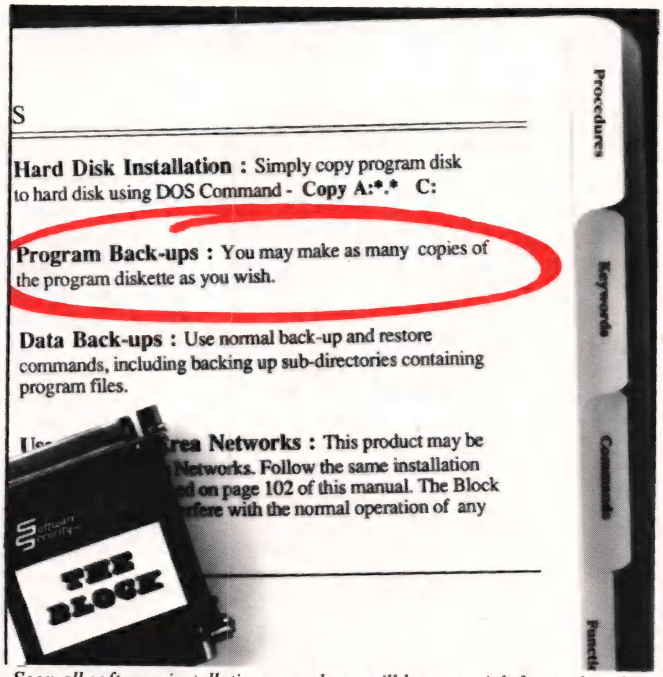
*Software protection must be understood to be a distinctively different concept from that commonly referred to as copy protection.*

Fundamentally, software protection involves devising a method that prevents unauthorized use of a program, without restricting a legitimate user from making any number of additional copies or preventing program operation via hard disk or LANs.

Logic dictates that magnetic media can no more protect itself from misuse than a padlock can lock itself.

Software protection must reside outside the actual storage media. The technique can then be made as tamper proof as deemed necessary. If one is clever enough, patent law can be brought to bear on the method.

Software protection is at a crossroads and the choices are clear. You can give product away to a segment of the market, or take a stand against the theft of your intellectual property.




*Soon all software installation procedures will be as straightforward as this. The only difference will be whether you include the option to steal your product or not.*

***"...giving your software away is fine..."***

We strongly believe that giving your software away is fine, if you make the decision to do so. However, if the public's sense of ethics is determining company policy, then you are no longer in control.

We have patented a device that protects your software while allowing unlimited archival copies and uninhibited use of hard disks and LANs. The name of this product is The BLOCK™.

The BLOCK is the only patented method we know of to protect your investment. It answers all the complaints of reasonable people concerning software protection.

In reality, the only people who could object are those who would like the option of stealing your company's product.

***"...eliminating the rationale for copy-busting..."***

Since The BLOCK allows a user to make unlimited archival copies the rationale for copy-busting programs is eliminated.

The BLOCK is fully protected by federal patent law rather than the less effective copyright statutes. The law clearly prohibits the production of work-alike devices to replace The BLOCK.

The BLOCK attaches to any communications port of virtually any microcomputer. It comes with a unique customer product number programmed into the circuit.

The BLOCK is transparent to any device attached to the port. Once it is in place users are essentially unaware of its presence. The BLOCK may be daisy-chained to provide security for more than one software package.

Each software developer devises their own procedure for accessing The BLOCK to confirm a legitimate user. If it is not present, then the program can take appropriate action.

***"...possibilities... limited only by your imagination..."***

The elegance of The BLOCK lies in its simplicity. Once you understand the principle of The BLOCK, hundreds of possibilities will manifest themselves, limited only by your imagination.

Your efforts, investments and intellectual property belong to you, and you have an obligation to protect them. Let us help you safeguard what's rightfully yours. Call today for our brochure, or a demo unit."

Software Security inc.

870 High Ridge Road Stamford, Connecticut 06905

**203 329 8870**

CIRCLE NO. 134 ON READER SERVICE CARD

# PANEL® *Plus*

## Advanced Screen Manager

Building an interactive application in C? **PANEL** *Plus* provides the features you need for professional program development:

**PRODUCTIVITY**

The **PANEL** *Plus* interactive screen design tools are the fastest way to get your application screens set out and tested. Just type prompts on the screen, mark out entry fields, define display and entry attributes, help boxes, borders, pop-up areas. Fields can be edited, moved and resized, and validation details entered – with the screen displayed so that you can see the effect of your changes.

Then **PANEL** *Plus* saves your work to disk, from where you can either load the design directly into your application program, or for better control, automatically generate C data structures, field areas, and header files which are compiled and linked into your program.

**QUALITY**

**PANEL** *Plus* screens can include all the features demanded by today's applications. Several different menu types are provided, including highlighted bars with help lines. Easy-to-use library functions support pop-up fields, horizontal and vertical scrolling in a field, and validation exits for supplied or custom data checking functions. Text functions can also be carried out in graphics mode using a supported graphics function library.

**EASE OF USE**

Although the library contains over 150 functions, it is logically organised so that most programs will only need to use a small subset. Users of Roundhill's PANEL package will find that the old calls have been emulated (and all screens are compatible). New, expanded documentation is provided with examples of all the main function calls. **PANEL** *Plus* includes full library source, with variant files for all supported systems, and no royalties are payable for the use of **PANEL** *Plus* libraries when linked into user applications.

**PORTABILITY**

**PANEL** *Plus* is designed to allow your programs to be ported to just about any environment where you can find a C compiler. Every version of **PANEL** *Plus* includes source modules for interfacing to all of the following environments: MS-DOS/PC-DOS (portable and memory-mapped), OS/2 protected mode, Amiga Intuition, Unix (with and without *termcap* or *termio*), Xenix, DOS-J (including Asian 16-bit character editing), and VAX/VMS. Graphics libraries supported include MetaWindow, HALO, Essential Graphics, and Microsoft C V5. The Microsoft mouse can be used in PC versions.

**PANEL** *Plus* for MS-DOS, with full library source, is priced at $495.00. Versions are available for the Manx Aztec, Borland, Lattice, MetaWare, Microsoft, and Wizard C compilers. Please call for prices of **PANEL** *Plus* for Xenix and Unix systems, and for VAX/VMS. Existing registered users of PANEL will receive a credit against the **PANEL** *Plus* license fee.

**New !**

Roundhill announces two screen tools for use with the exciting new C compilers from Borland and Microsoft:

**PANEL/TC – $129.00**
*For use with Borland's Turbo C*

**PANEL/QC – $129.00**
*For use with Microsoft's Quick C*

Each package is configured for use on an IBM PC or compatible system, and screens can be designed and built into your programs while running in the special development environment provided with the compiler.

All the **PANEL** *Plus* library functions are supported, and source code for every validation function is provided so that you can customise the entry checking to suit your application. When you need to move your programs to other environments, or need the full library source for other reasons, upgrades to **PANEL** *Plus* are available.

PANEL/QC can be run in any of the graphics modes supported by Quick C, and also interfaces to Microsoft C V5.

**Roundhill Computer Systems Limited** **(201) 569 2265**
**PO Box 8107, Englewood NJ 07631**

**Roundhill Computer Systems Limited** **(0672) 54675**
**PO Box 14 Marlborough SN8 1LR England**

**Telex (UK): 444453 AWARE G** **BIX:** *join roundhill*
**Fax (UK): (0672) 54436**

# Roundhill Computer Systems

CIRCLE NO. 135 ON READER SERVICE CARD

**Listing One** *(Listing continued, text begins on page 30.)*

```
      25: TriggerEnabled := True; {enable trigger}

      26: BEGIN   {Stop Triggering}
             TriggerEnabled := False;
             {stop triggering for both channels}
             COM1_Is_Triggered := False;
             COM2_Is_Triggered := False;

             {start data display for both channels}
             COM1_Display_Data := True;
             COM2_Display_Data := True;
             GoToXY(3,25);
             ClrEol;
             WriteString('Triggering Disabled',Low);
          END;

      27: BEGIN   {Ascii Normal Data Display}
             AsciiDisplay := True;
             HandShakeDisplay := False;
          END;

      28: BEGIN   {Ascii with handshake Data Display}
             AsciiDisplay := True;
             HandShakeDisplay := True;
          END;

      29: BEGIN   {Hex Normal Data Display}
             AsciiDisplay := False;
             HandShakeDisplay := False;
          END;

      30: BEGIN   {Hex with handshake Data Display}
             AsciiDisplay := False;
             HandShakeDisplay := True;
          END;

      31: BEGIN   {Toggle adding spaces to display}
             AddSpace := NOT AddSpace;
          END;

      32: BEGIN   {Start data acquire}
             COM1_Data_Acquire := True;
             COM2_Data_Acquire := True;
          END;

      33: BEGIN   {Stop data acquire}
             COM1_Data_Acquire := False;
             COM2_Data_Acquire := False;
          END;

      34: BEGIN   {Clear the display}
             {claim the screen}
             Alloc(ScreenAccess);
             ClrScr;
             DrawMenuFrame;
             {set cursor postion to home}
             OldXPos := 1;
             OldYPos := 1;
             Dealloc(ScreenAccess);
          END;

      35: Init_Program; {reset the analyzer}

      36: BEGIN   {End the analyzer program}
             ExitMenu := True;
             ExitProgram := True;
          END;


    END;
  END;
END;
{************ Begin Task Procedures *************}

PROCEDURE ProcessKeysTask;

VAR

 Ch:   Char;

 Done,
 OldCOM1Flag,
 OldCOM2Flag:  Boolean;

BEGIN

   Done := False;

   REPEAT

      {read key if available else yield}

      Ch := Char(GetKey);

      CASE Ch OF

      Esc: BEGIN  {if key is Esc}
             {display and process the menu}
             Done := DoMenu;
             {rebuild display when finished}
             BuildDisplay;
           END;

      Sp: BEGIN  {if space bar}
             {save old state}
             OldCOM1Flag := COM1_Display_Data;
             OldCOM2Flag := COM2_Display_Data;

             {stop filling of the display fifo so}
             {it does not overflow durin pause}
             {turn off display data both channels}
             COM1_Display_Data := False;
             COM2_Display_Data := False;
             GoToXY(1,24);
             ClrEol;
             GoToXY(23,24);
             WriteString(
               'Press <Enter> to restart the display'
               ,HighBlink);
             REPEAT
                {loop until Cr starts display again}
                Ch := Char(GetKey);
             UNTIL Ch = Cr;

            {restore previous display status}
            COM1_Display_Data := OldCOM1Flag;
            COM2_Display_Data := OldCOM2Flag;
            BuildDisplay; {rebuild display}
          END;
      END;

   UNTIL Done;

END;


PROCEDURE MoveCOM1Data;

{Move data from the COM1 input buffer to the}
{COM2 output buffer and to the display buffer}
{if enabled}

VAR

 Sd:   DataRec;
 Dd:   DisplayRec;

BEGIN

   REPEAT
      {yield if no data to move}
      WHILE COM1_Input_Fifo.Ovd.Count = 0 DO
         yield;
      {when data is available move it}

      {turn ints off, read data, turn ints on}
      INLINE($FA);
      GetSerialData(COM1_Input_Fifo,Sd);
      INLINE($FB);

      {store in COM2 output fifo}
      PutSerialData(Sd,COM2_Output_Fifo);
      {if a trigger is enabled}
      IF COM1_Is_Triggered AND
         TriggerEnabled    THEN
      BEGIN
         {and a match is found}
         IF Sd.Data = TriggerPattern THEN
         {indicate match & disable further matches}
         {trigger is single shot event}
         BEGIN
            TriggerEnabled := False;

            WriteStringAt('   COM1 Triggered
                          ,Low,3,25);
           {If displaying before trigger}
           {then stop data display}
           IF TriggerMode = Before THEN
               COM1_Display_Data := False
           ELSE
               {start the data display}
               COM1_Display_Data := True;

         END;
      END;
```

*(continued on next page)*

**Listing One** *(Listing continued, text begins on page 30.)*

```
      {if we are displaying data}
      IF COM1_Display_Data THEN
      {move data to the display fifo also}
      BEGIN
         {tagged from this COM device}
         Dd.DR := Sd;
         Dd.Tag := FromCOM1;
         PutDisplayData(Dd,DisplayFifo);
      END;

   UNTIL False;

END;


PROCEDURE MoveCOM2Data;

{Move data from the COM2 input buffer to the}
{COM1 output buffer and to the display buffer}
{if enabled}

VAR

 Sd:   DataRec;
 Dd:   DisplayRec;

BEGIN

   REPEAT
      {yield if no data to move}
      WHILE COM2_Input_Fifo.Ovd.Count = 0 DO
          yield;

      {when data is available move it}

      {ints off, read data, turn ints on}
      INLINE($FA);
      GetSerialData(COM2_Input_Fifo,Sd);
      INLINE($FB);

      {store data in COM1 output fifo}
      PutSerialData(Sd,COM1_Output_Fifo);

      {if a trigger is enabled}
      IF COM2_Is_Triggered AND
         TriggerEnabled    THEN
      BEGIN
         {indicate match & disable further matches}
         {trigger is single shot event}
         IF Sd.Data = TriggerPattern THEN
         BEGIN
            TriggerEnabled := False;
            WriteStringAt('   COM2 Triggered
                          Low,3,25);
            {If displaying before trigger}
            {stop data display}
            IF TriggerMode = Before THEN
               COM2_Display_Data := False
            ELSE
               {If displaying after trigger}
               {start data display}
               COM2_Display_Data := True;

         END;
      END;
      {if we are displaying data}
      IF COM2_Display_Data THEN
      BEGIN
         {move data to the display fifo also}
         Dd.DR := Sd;

         {tagged from this COM device}
         Dd.Tag := FromCOM2;
         PutDisplayData(Dd,DisplayFifo);
      END;

   UNTIL False;

END;


PROCEDURE OutputCOM1Data;

{Move serial data from the output fifo}
{to the COM port}
```

*(continued on page 62)*

*"Developing my application in C would have taken 6 months to a year, but in Actor it took 2 months."* —*Brian Fenske, Boeing Commercial Airplane Company*

# "To C or not to C..."


***Actually***, you don't have to make the choice. Once C was ideal for all PC programming. But it has been complicated by windowing and graphical interfaces. Now windows development with C is difficult, time-consuming and error-prone. You need a new language that simplifies windows programming. Introducing Actor®.

***Actor*** is the first interactive object-oriented language made for commercial development. Its powerful browsers, inspectors and debuggers give you more insight into a windowing environment than C ever will. But your C work is not lost. C libraries can be linked to Actor. Plus, its procedural syntax is easy for C programmers to learn.

***Actor*** comes with windowing classes built in. Customize Actor's classes to create stand-alone windowing applications. And objects give you another layer of independence for a smooth transition to OS/2 and Presentation Manager. It's the quickest and easiest way to write a windowing program.

*"You can write Windows programs much faster with Actor than with C or assembly language."*
*—PC Magazine, June 9, 1987*

## Tech Specs

New Release Version 1.1

- Runs with Microsoft Windows 1.04, 2.0 and 386. Extended memory under 2.0 and 386.
- Pure, single-inheritance object-oriented language, incrementally compiled.
- Dynamic linking to C, Pascal, Assembler, or Fortran libraries. Pass data in C structures.
- Pascal and C-like syntax.
- Programming tools: Browser, Inspector, Debugger, File Editor.
- Full access to MS-Windows systems calls, multitasking, and DDE.
- Fast device-independent graphics: lines, shapes, icons, cursors, bitmaps, metafiles, Turtle graphics, sample control language using YACC.
- 150 classes, 1500 functions, fully extensible.
- Window styles: tiled, overlapping, popup, child, edit, dialogs. Controls: list boxes, scroll bars, buttons, check boxes.
- Data structures: stacks, arrays, queues, lists, dictionaries, sets, sorting, hashing, intervals.
- AI support: frames, symbols, dictionaries, lists, symbolic programming, functional arguments. Parsing and lexical analysis YACC compatible.
- String manipulation: substring, concat, append, insert, remove, search.
- 643-page manual includes tutorial and reference.
- No license fees. Generates stand-alone applications.
- Fastest interactive OOL available.
- Fast incremental garbage collector.

**Actor $495 • Academic price $99 • Academic site license $99 • Manuals for site license $35 • New! Language Extension $99 • Shipping $5 US, $25 Int'l**

**The Whitewater Group**
**Technology Innovation Center**
**906 University Place, Evanston, Illinois 60201**
**(312) 491-2370**

Actor is a registered trademark of The Whitewater Group, Inc.

CIRCLE NO. 137 ON READER SERVICE CARD

**Listing One** *(Listing continued, text begins on page 30.)*

```
CONST

 {max of 80 chars to serial port}
 {between yields to other tasks}

 OutputsPerYield = 80;


VAR

 Outs:    Integer;
 Sd:      DataRec;
 Urgent: Boolean;

BEGIN

   REPEAT

      {do until data is available in fifo}
      WHILE (COM1_Output_Fifo.Ovd.Count = 0) DO
      BEGIN
         {always make the handshake lines}
         {between COM1 and COM2 agree then yield}
         MakeHandshake(COM1,COM2_Status);
         yield;
      END;

      {we have data to output}
      {Its urgent if fifo is over half full}

      Urgent := (COM1_Output_Fifo.Ovd.Count >=
                    (SerialDataFifoSize/2));

      {Initialize output counter to max value}
      Outs := OutputsPerYield;

      {While there is data to output}

      WHILE ((COM1_Output_Fifo.Ovd.Count <> 0) AND
             (Outs <> 0)) DO
      BEGIN

         {test if UART is ready to transmit}

         WHILE (Get_Serial_Status(COM1) AND
                TxRdyBit) = 0 DO
         BEGIN
            {if we have time yield until ready}
            IF NOT Urgent THEN
               yield;
         END;

         {get data record to output}
         {set the handshake lines and output}
         {the data}

         GetSerialData(COM1_Output_Fifo,Sd);
         MakeHandshake(COM1,Sd.Status);
         port[COM1] := Sd.Data;

         {one less to output}
         Outs := Outs - 1;

      END;
      yield;

   UNTIL False;

END;
PROCEDURE OutputCOM2Data;

{Move serial data from the output fifo}
{to the COM port}

CONST

 {max of 80 chars to serial port}
 {between yields to other tasks}

 OutputsPerYield = 80;


VAR

 Outs:    Integer;
 Sd:      DataRec;
 Urgent: Boolean;

BEGIN

   REPEAT

      {do until data is available in fifo}
      WHILE (COM2_Output_Fifo.Ovd.Count = 0) DO
      BEGIN
         {always make the handshake line}
         {between COM1 and COM2 agree then yield}
         MakeHandshake(COM2,COM1_Status);
         yield;
      END;

      {we have data to output}
      {Its urgent if fifo is over half full}

      Urgent := (COM2_Output_Fifo.Ovd.Count >=
                 (SerialDataFifoSize/2));

      {Initialize output counter to max value}
      Outs := OutputsPerYield;

      {While there is data to output}

      WHILE ((COM2_Output_Fifo.Ovd.Count <> 0) AND
             (Outs <> 0)) DO
      BEGIN
         {test if UART is ready to transmit}

         WHILE (Get_Serial_Status(COM2) AND
                TxRdyBit) = 0 DO
         BEGIN
            {if we have time yield until ready}
            IF NOT Urgent THEN
               yield;
         END;

         {get data record to output}
         {set the handshake lines and output}
         {the data}

         GetSerialData(COM2_Output_Fifo,Sd);
         MakeHandshake(COM2,Sd.Status);
         port[COM2] := Sd.Data;

         {one less to output}
         Outs := Outs - 1;

      END;
      yield;

   UNTIL False;

END;


PROCEDURE DisplayCOMData;

CONST

 {max # of items displayed before yielding}

 MaxDisplayItems = 20;

VAR

 DisplayItems,
 DataLen:       Integer;
 D:             DisplayRec;
 BEGIN

    {initial cursor is homed}
    {in data display window}
    OldXPos := 1;
    OldYPos := 1;

    REPEAT
       {if data in display fifo}
       IF DisplayFifo.Ovd.Count <> 0 THEN
       BEGIN
          {claim the screen, define the window}
          {position the cursor and initialize}
          {items counter}
          Alloc(ScreenAccess);
          Window(1,7,80,23);
          GoToXY(OldXPos,OldYPos);
          DisplayItems := MaxDisplayItems;

          WHILE ((DisplayFifo.Ovd.Count <> 0 ) AND
                 (DisplayItems <> 0)) DO
          {while there is data to display}
          BEGIN
             {get the data, display it, dec}
             {item counter. Save len of displayed}
             {item.}

             GetDisplayData(DisplayFifo,D);
             DataLen := DisplayData(D);
             DisplayItems := DisplayItems - 1;
```

*(continued on page 65)*


# Our software comes with something no one else can offer.

## Lattice Service.

When you join the Lattice family of customers, you'll discover that your software purchase is backed by more than just an excellent warranty. It's backed by unparalleled technical support. By a total commitment to your success and satisfaction. And by Lattice's dedication to excellence in products and services.

Unlike other software manufacturers who charge you for services after you've purchased their product, Lattice offers a unique package of support programs at a price we can all live with—FREE.

**Lattice Bulletin Board Service**
LBBS is our 24-hour a day bulletin board system that allows you to obtain notification of new releases, general information on Lattice products, and programs for the serious user. And if you've ever experienced the frustration of having to wait a year or more for a new release (that has corrected a bug), you'll really appreciate LBBS. Because with this service, you can actually download the latest program fixes to instantly eliminate any bugs discovered after release.

**Technical Support Hotline**
Responsible, dependable and capable Support Representatives are only a phone call away. You will talk to a highly skilled expert who is trained to answer any questions you have relating to specific Lattice products. Remember, your complete satisfaction is our goal.

**McGraw-Hill BIX™ Network**
The Byte Information Exchange (BIX) Network is a dial-in conference system that connects you with a Special Interest Group of Lattice users. The nominal one-time registration fee allows you to BIX-mail your questions—via your modem—directly to Lattice. Or you can post your questions in the conference mode for Lattice or other users to answer. Once again, you have 24-hour access.

**You Also Receive:**
- Timely updates and exciting enhancements
- 30-day, money-back guarantee
- Lattice Works Newsletter
- Technical Bulletins
- Access to Lattice User Groups

Lattice has developed more than 50 different Microcomputer software tools that are used by programmers worldwide. We were there for every MS-DOS release. We're there now for OS/2. And we'll be there for the next generation of technical changes. But most of all, Lattice is there for you.


Available through dealers and distributors worldwide.

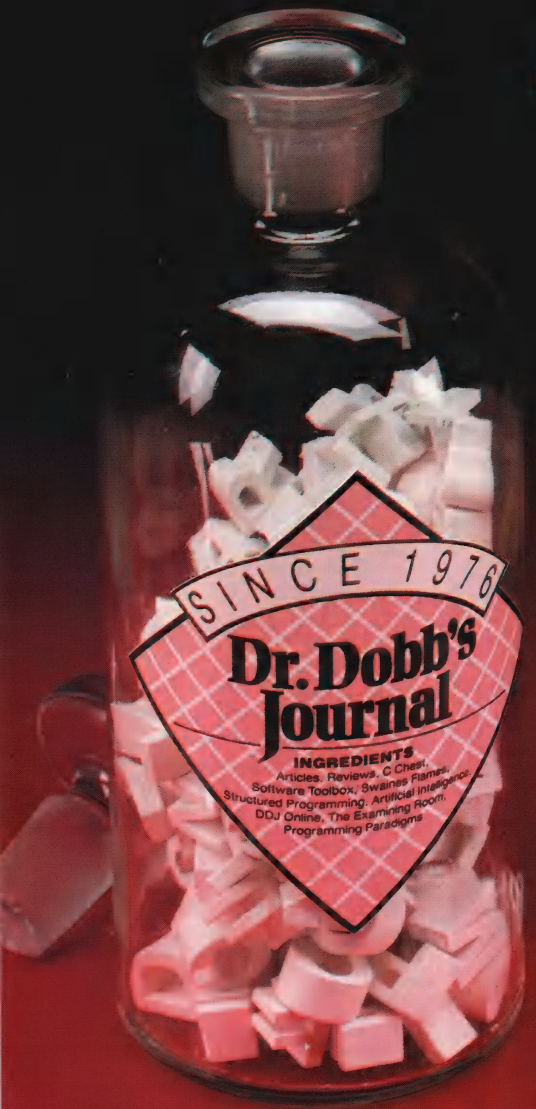
Lattice

Lattice, Incorporated
2500 S. Highland Avenue
Lombard, IL 60148
Phone: 800/533-3577
In Illinois: 312/916-1600

Subsidiary of SAS Institute Inc.

CIRCLE NO. 138 ON READER SERVICE CARD

# THE CURE FOR COMMON CODE




Are you getting the recommended monthly allowance of C, Assembly, Forth, Pascal, BASIC or Modula-2? Subscribe to *Dr. Dobb's Journal of Software Tools* and you won't catch any nasty bugs again!

Each month the Doctor brings you aid for ailing algorithms and the cure for common code. For the latest developments in software design and pages of code that will make you a more productive programmer, take the Dr. Dobb's prescription.

For more than a decade, the programming elite have known *Dr. Dobb's Journal* to be the foremost source of software tools. Subscribe now and get your monthly dose from the Doctor.

# Dr. Dobb's Journal of Software Tools

# The Rx for Programmers

## Subscribe Now & Save Over 15% Off the Newsstand Price!




$5 SAVINGS

## SUBSCRIBE AND SAVE!
Subscribe to
## DR. DOBB'S JOURNAL OF SOFTWARE TOOLS
and save over $5—a 15% savings off the cover price!

☐ 1 year $29.97 ☐ 2 years $56.97

☐ Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment enclosed ☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

ONLY $29.97! YOU SAVE OVER $5.00

*Savings based on the full one-year newsstand rate of $35.40. All foreign countries please add $11 per year for surface mail; Canada & Mexico add $28 per year for airmail; other countries add $32 per year for airmail. All foreign subscriptions must be paid in U.S. funds drawn on a U.S. bank. Please allow 6-8 weeks for delivery.

A Publication of M & T Publishing, Inc.

361S


## COMMENTS & SUGGESTIONS

**Dear Reader,** February 1988, #136

Dr. Dobb's has a long tradition of listening to its readers. We like to hear when something really helps or, for that matter, bothers you. In this hectic world of ours, however, it is often difficult to take time to write a letter. This card provides you with an easy way to correspond and, if you include your name and address, we may use appropriate comments in The Letters column. Simply fill it out and drop it in the mail. —Ed

Which articles or departments did you enjoy the most this month? Why?

_____

_____

_____

Comments or suggestions _____

_____

_____

_____

Name: _____

Address: _____

**Listing One** *(Listing continued, text begins on page 30.)*

```
                {if not room to display for another}
                {item of same length and we're on the}
                {last display line in window, then}
                {home cursor and start overwriting}
                {screen so as to avoid the scroll of}
                {the display. If not on last line of}
                {window, advance to next line. Either}
                {way, the next line must be cleared.}

                IF WhereX + DataLen >= 80 THEN
                   IF WhereY = 17 THEN
                      GoToXY(1,1)
                   ELSE
                      WriteLn;

                ClrEol;
             END;

             {save new cursor position}
             OldXPos := WhereX;
             OldYPos := WhereY;

             {back to full screen}
             {and release screen lock and yield}
             Window(1,1,80,25);
             Dealloc(ScreenAccess);
          END
          ELSE
             {if no data to display just yield}
             yield;

       UNTIL False;

    END;


    PROCEDURE Timer;

    VAR

     Toggle:  Boolean;

    BEGIN

    Toggle := True;

    REPEAT

       {every 1/2 second do the following}
       pause(4);

       {if ok to update the screen}
       IF UpdateScreenStatus THEN
       BEGIN
          {lock the screen}
          Alloc(ScreenAccess);

          {if status of either port has changed}
          IF ((COM1_Status <> OldCOM1_Status)  OR
              (COM2_Status <> OldCOM2_Status)) THEN

          {then update the screen}
          BEGIN
             {display new COM1 status}
             DisplayHandShakeStatus(COM1,COM1_Status
                                    ,COM2_Status);
             {save new status}
             OldCOM1_Status := COM1_Status;

             {display new COM2 status}
             DisplayHandShakeStatus(COM2,COM2_Status
                                    ,COM1_Status);
             OldCOM2_Status := COM2_Status;

          END;

          {every full second update}
          {buffer status on screen}
          IF Toggle THEN
             DisplayBufferStatus;

          {unlock the screen}
          Dealloc(ScreenAccess);
          Toggle := NOT Toggle;
       END;

    UNTIL False;

END;
```

*(continued on page 46)*

## Listing One *(Listing continued, text begins on page 30.)*

```
BEGIN {main}

    {verify presence of COM1 and COM2}
    VerifyHardware;
    {initialize the multitasker}
    Init_Kernel;
    {initialize the menu structure}
    Init_Menu;
    {install the serial ISRs}
    Install_Serial_Handlers;

    {Initialize strings used to format the display}
    Line1 := Char(201)+' COM1 '+repl(16,Char(205))+
           ' Serial Protocol Analyzer Ver: 1.0 '+
           repl(15,Char(205))+ ' COM2 '+Char(187);

    Line2 := Char(186)+' In Fifo:'+repl(5,' ')+
             'Out Fifo:    '+ Char(186)+
             repl(10,' ')+'By'+repl(10,' ')+
             Char(186)+' In Fifo:'+repl(5,' ')+
             'Out Fifo:    '+Char(186);

    Line3 := Char(186)+' Stat- BRK:  FE:  PE:  OR: '+
             Char(186)+ '   Craig A. Lindley   '+
             Char(186)+ ' Stat- BRK:  FE:  PE:  OR: '+
             Char(186);

    Line4 := Char(186)+' In - CD:  RI:  DSR:  CTS: '+
             Char(186)+repl(22,' ')+Char(186)+
             ' In - CD:  RI:  DSR:  CTS: '+Char(186);

    Line5 := Char(186)+ ' Out- DTR:      RTS:'+
             repl(7,' ')+Char(186)+
             '   Display Fifo:      '+Char(186)+
             ' Out- DTR:      RTS:'+repl(7,' ')+
             Char(186);

    Line6 := Char(200)+repl(27,Char(205))+
             Char(202)+repl(22,Char(205))+
             Char(202)+repl(27,Char(205))+
             Char(188);

    Line24 := '  <Esc> for Menu  <Space Bar> to Pause'+
              '    COM1 - Normal : COM2 - Reverse Video';

    BuildDisplay;   {show main display}
    Init_Program;   {initialize all vars}

    {initialize screen lock}
    Initialize_Semaphore(ScreenAccess);

    {enable status update}
    UpdateScreenStatus:= True;

    {Fork off all tasks}
     Fork;

     IF child_process THEN
        MoveCOM1Data;

     Fork;

     IF child_process THEN
        MoveCOM2Data;

     Fork;

     IF child_process THEN
        OutputCOM1Data;

     Fork;

     IF child_process THEN
        OutputCOM2Data;

     Fork;

     IF child_process THEN
        DisplayCOMData;

     Fork;

     IF child_process THEN
        Timer;


     ProcessKeysTask;

     ClrScr;
     Disable_Serial_Devices;
     Remove_Serial_Handlers;

END
```

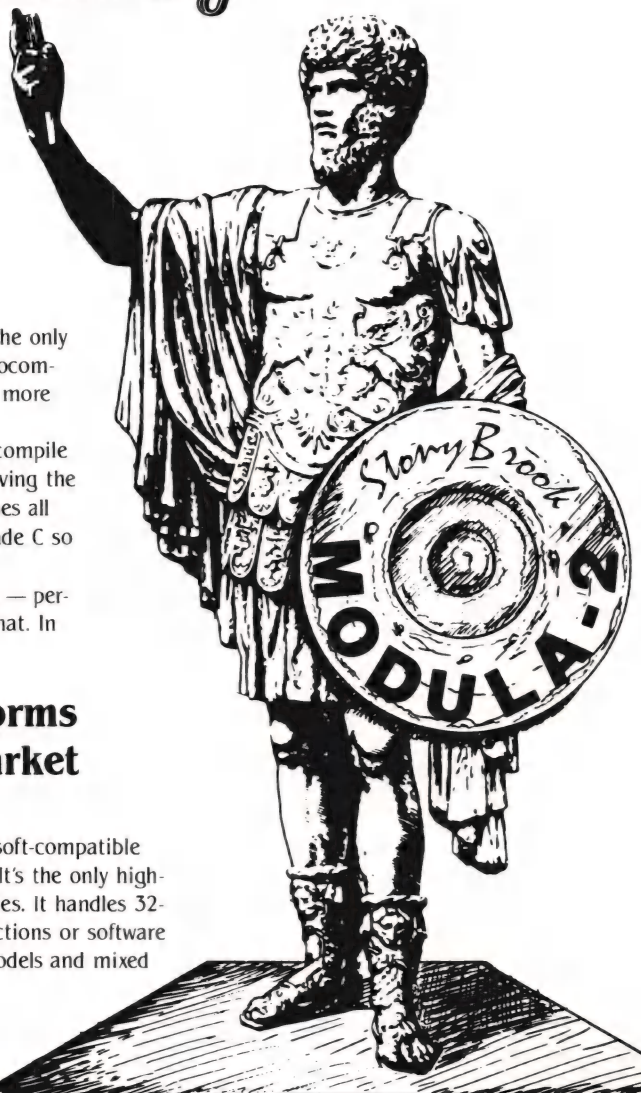
**End Listing One**

## Listing Two

```
{***********************************************}
{***                                         ***}
{***       Menu System support procedures    ***}
{***      for the serial protocol analyzer   ***}
{***                written by               ***}
{***             Craig A. Lindley            ***}
{***                                         ***}
{***    Ver: 2.0     Last update: 08/15/87   ***}
{***                                         ***}
{***********************************************}

CONST

 {max # of video attributes - 1}
 AttribMax = 5;

 Attributes: ARRAY[0..AttribMax] OF
 RECORD
    f,              {foreground color}
    b: Integer;     {background color}
 END

 {Attributes are:}
 {Low, High, Rev, LowBlink, HighBlink, RevBlink}

  = ((f:7; b:0) , (f:15;b:0) , (f:0; b:7),
     (f:23;b:0) , (f:31;b:0) , (f:16;b:7));


 {menu display lines relative to screen line 1}
 MenuLine1 = 2;
 MenuLine2 = 3;
 MenuLine3 = 4;
 MenuLine4 = 5;

 {max A tree dimensions}
 level2width = 7;
 level3width = 5;
 level4width = 7;

 home=    #199;    {home key code}
 larrow=  #203;    {left cursor arrow code}
 rarrow=  #205;    {right cursor arrow code}
 endkey=  #207;    {end key code}
 bs=      #08;     {backspace key}
 lf=      #10;     {line feed code}
 cr=      #13;     {carrage return code}
 Esc=     #27;     {escape key code}
 sp=      #32;     {ascii space code}

TYPE

 AttribType  = (Low ,High, Rev, LowBlink,
                HighBlink, RevBlink);

 {data structure for atree menu entry}
 {see text for details}

 menu_entry=
 RECORD
    title:  STRING[10];
     desc:  STRING[40];
    chars:  STRING[8];
    index:  Byte;
    ccode:  Byte;
 END;

tree = ARRAY[0..level2width,
             0..level3width,
             0..level4width]
       OF menu_entry;

VAR

 ExitMenu,
 ExitProgram:      Boolean;

 ind1,ind2,
 ind3,
 selector,
 cmd_code,
 level:            Byte;

 {atree data structure used for nested menus}
 atree:            tree;


{******* Keyboard and Display Procedures ********}

PROCEDURE Beep;
```

*(continued on page 68)*

# I have come to bury C, sir, not to praise it.




C served us well in the days of kilobytes and kilohertz. It was the only language we could implement efficiently on our newborn microcomputers. But with today's mega-machines, shouldn't we demand more from our compilers?

Modula-2 increases productivity by catching your errors at compile time. You'll easily modularize and structure your programs, driving the hordes of barbaric bugs into the hinterlands. And Modula-2 does all this without taking away the low-level machine access that made C so popular.

Until now, you had to pay a price for the Modula-2 advantages — performance just didn't measure up to C. But we've changed all that. In a suite of benchmarks developed by PC Week:

## Stony Brook Modula-2 outperforms the best C compilers on the market

(and no other Modula-2 compiler even comes close).

Stony Brook Modula-2, for 80x86 machines, produces Microsoft-compatible objects, and fully supports both Microsoft Windows and OS/2. It's the only high-level language compiler that lets you write dynamic link libraries. It handles 32- and 64-bit real numbers with in-line 80x87 coprocessor instructions or software emulation. And Stony Brook Modula-2 supports six memory models and mixed model programming.

You might want to bury your C compiler once you have used Stony Brook Modula-2, but you won't have to. We made it possible to directly call C and other languages from Modula-2, so you won't have to throw away your investment in C code.

So, friends, programmers, and C-users, lend us your ears. Call us or write for more information and to find out how you can get a demo compiler.

| Benchmark | Stony Brook M-2 V1 | Logitech M-2 V3 | Turbo C V1 | Microsoft C V4 |
|---|---|---|---|---|
| Magnifier | 0.05 | 0.06 | 0.05 | 0.05 |
| For Loop | 0.28 | 0.73 | 0.44 | 0.68 |
| While Loop | 0.35 | 0.75 | 0.49 | 0.66 |
| Repeat Loop | 0.33 | 0.68 | 0.38 | 0.66 |
| Literal Assign | 0.33 | 0.97 | 0.60 | 0.94 |
| Memory Access | 0.33 | 1.04 | 0.55 | 1.00 |
| Real Arithmetic | 4.68 | 13.40 | 3.73 | 2.96 |
| Real Algebra | 4.18 | 12.87 | 3.07 | 3.64 |
| Vector | 0.77 | 1.75 | 1.21 | 1.26 |
| Equal If | 0.66 | 1.47 | 0.93 | 1.32 |
| Unequal If | 0.66 | 1.43 | 0.99 | 1.32 |
| No Parameters | 0.33 | 0.60 | 0.34 | 1.04 |
| Value | 0.50 | 0.71 | 0.60 | 1.21 |
| Reference | 0.61 | 0.77 | 0.66 | 1.21 |
| Real Maths | 3.90 | 4.61 | 7.85 | 18.51 |

The compiler package includes DOS runtime library objects and full sources for our split-screen text editor for $195. The development package includes all of the above plus an automatic make utility, a symbolic debugger, and run-time library sources for $345. MasterCard and Visa accepted. Add $5 for shipping in North America, $25 for overseas shipping.

Stony Brook INC. SOFTWARE

Forest Road, Wilton, New Hampshire 03086 ● (603)654-2525 Ask for Cleopatra.

CIRCLE NO. 139 ON READER SERVICE CARD

**Listing Two** *(Listing continued text begins on page 30.)*

```
BEGIN

   Sound(1000);
   Pause(1);
   NoSound;

END;


FUNCTION GetKey : Byte;

VAR

 Ch: Char;
 B:  Byte;

 CurrentX,
 CurrentY:  Integer;

BEGIN

  {save cursor position}
  {because we are going to yield}
  {and loose control of display}

  CurrentX := WhereX;
  CurrentY := WhereY;

  {yield until a key is pressed}

  WHILE NOT keypressed DO
      yield ;

  {put the cursor back}
  GoToXY(CurrentX,CurrentY);

  {read the key}
  read(kbd,Ch);

  {see if its an extended key}
  IF (Ch = Esc) AND keypressed THEN
  BEGIN
    {if so read again and mark}
    {as extended by setting MSB}
    read(kbd,Ch);
    B := ord(Ch)+128;
  END
  ELSE
    {in either case B has key code}
    B := ord(Ch);

  GetKey := B;

END;


FUNCTION repl (Count:Integer; ch:Char):FullString;

{replicate a char into a string of}
{specified length}

VAR

 i:               Integer;

BEGIN

   FOR i:=1 TO Count DO repl[i]:=ch;
   repl[0]:=Char(Count);

END;


PROCEDURE WriteString (Stng:FullString;
                       Attrib:AttribType);

BEGIN

   {set the foreground and background}
   {colors and write the string}

   WITH Attributes[ORD(Attrib)] DO
   BEGIN
      TextColor(f);
      TextBackGround(b);
   END;
   write(Stng);

   {go back to low video mode}
   WITH Attributes[ORD(Low)] DO
   BEGIN
      TextColor(f);
      TextBackGround(b);
   END;

END;


PROCEDURE WriteStringAt (Stng:FullString;
                         Attrib:AttribType;
                         X,Y:Integer);

BEGIN

 GoToXY(X,Y);
 WriteString(Stng,Attrib);

END;


{********** Start of Menu Procedures **********}

PROCEDURE center (Line,Width:Integer;
                  Outstr:FullString;
                  Attrib:AttribType);

{Center and write string with attribute on a}
{specified line in a given width field}

BEGIN

   GoToXY(1,Line);
   Clreol;
   GoToXY((Width DIV 2)-(length(Outstr) DIV 2),
          Line);
   WriteString(Outstr,Attrib);

END;


PROCEDURE DrawFrame (x,y,width,height:Integer);

VAR

 top,bottom:     Str80;

BEGIN

   top:=   repl(width,Char(205));
   bottom:=repl(width,Char(205));
   WriteStringAt(top,High,x,y);
   WriteStringAt(bottom,High,x,y+height-1);

END;


PROCEDURE DrawMenuFrame;

BEGIN

   DrawFrame(1,1,80,6);
   center(MenuLine1,78,
          ' Serial Protocol Analyzer Menu ',Rev);

END;


PROCEDURE DisplayMenu (ind1,ind2,ind3,level,
                       selector:Byte);

VAR

 tind1,tind2,
 tind3,i:      Integer;
 attrib:       AttribType;
 titlestr:     FullString;

BEGIN
   titlestr:='-- '+
     atree[ind1,ind2,ind3].title+' Selection'+' --';
   center(MenuLine2,78,titlestr,Low);
   GoToXY(1,MenuLine4);
   clreol;
   GoToXY(9,MenuLine3);
   clreol;

   FOR i:= 1 TO atree[ind1,ind2,ind3].index DO
   BEGIN
      CASE level OF
         1: ind1:=i;
         2: ind2:=i;
```

*(continued on page 72)*

# Clarify and document your source listing and get an "organization chart" of your program's structure

## with two NEW utilities from Aldebaran Laboratories, for C, BASIC, Pascal, dBASE,® FORTRAN and Modula-2 programmers.

Now works with FORTRAN

**"Occasionally, a utility comes along that makes a programmer's life much easier. SOURCE PRINT is such a program. It contributes to the programmer's job by organizing code into a legible format and by helping to organize the documentation and debugging process."**

**— PC Magazine**
**Sept. 16, 1986**

Source Print and Tree Diagrammer both have easy-to-use menus with point-and-shoot file selection, and let you search for files containing a given string. For IBM PC and compatibles with 256K.

Join thousands of programmers who are working more efficiently using Source Print and Tree Diagrammer. Order these indispensable tools today. We ship immediately, and there's no risk with our 60-day money-back guarantee. **Order both and save. Only $155.00.**

800-257-5773 Dept. 58
In California:
800-257-5774 Dept. 58

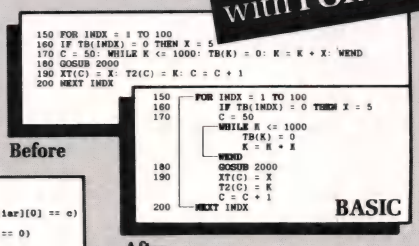
MasterCard, VISA, American Express, COD. Add $5 for shipping/handling.

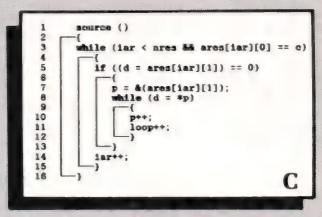
or see your local dealer!

Source Print and Tree Diagrammer are trademarks of Aldebaran Labs. dBASE is a trademark of Ashton Tate. Prices subject to change without notice.

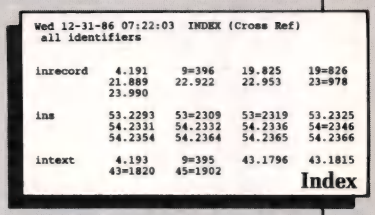
## Source Print™

**organizes your source code, simplifies debugging, and makes documentation a snap!** It lists one or more source files with informative page headings and optional line numbers, while offering invaluable features:




**The Index** (Cross-Reference list) saves you time by showing exactly where variables are used and where functions, procedures, and routines are called.



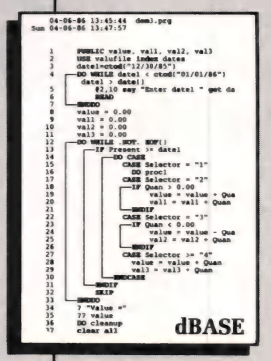

**$97.00**

Locations where new values may be assigned to variables are shown, making it easy to track down that mysterious value change.

**Structure Outlining** solves the problem of hard-to-see nested control structures by automatically drawing lines around them.

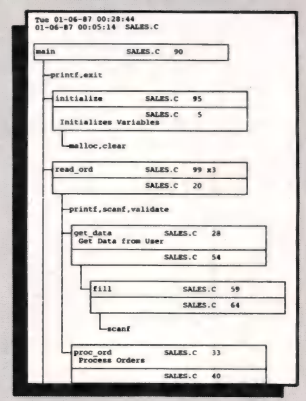
**Automatic Indentation** of source code and listings reduces your editing time and ensures indentation accuracy.

**Plus** . . . Source Print generates a table of contents listing functions and procedures. Keywords can be printed in boldface on most printers. Multi-statement BASIC lines can be split for readability. Functions and procedures can be drawn by name from one or more source files to form a new file.




## Tree Diagrammer™

**shows your program's overall organization at a glance.** Ordinary program listings merely display functions, procedures, and subroutines sequentially, but do not display the relationships between these routines. Our revolutionary new Tree Diagrammer automatically creates an "organization chart" of your program showing the hierarchy of calls to functions, procedures, and subroutines. Recursive calls are indicated and designated comments in the source code will appear on the chart.



Tree Diagrammer helps you organize your program more logically. And you'll be amazed at how easy it is to debug when you see how your routines interact.

**$77.00**

---

**Aldebaran Laboratories** 3339 Vincent Rd. Pleasant Hill, CA 94523 415-930-8966

YES! Rush me ☐ Source Print @ $97. _____ ☐ Tree Diagrammer @ $77. _____
☐ Both $155. Ship/Handling $5. For CA add 6% tax _____ Total _____
Name _____
Company _____
Address _____
City _____ State _____ Zip _____
☐ Check enclosed ☐ VISA ☐ MasterCard ☐ American Express
Card # _____ Exp. Date _____
Signature _____ Phone # _____ 58

CIRCLE NO. 140 ON READER SERVICE CARD

# THE PROGRAMMER'S SHOP

**helps save time, money, and cut frustrations. Compare, evaluate, and find products.**

## Your Complete Tool Set: Only $50 per Month!

You could easily spend $1,500 pulling together the best programming environment. Now you can lease your choice of the best compiler, editor, debugger, and general purpose library, or whatever.

If your 2-year-old business generates $75K+ per year in revenue, call about our software-only lease.

*Another innovation from The Programmer's Shop.*

**Call a Software "Leasing Specialist."**

## C Programmers: Complete Your Development Toolset.

Dbase translation, fussy preprocessing, graphics windowing, ISAMs: C tools cover the spectrum.

Select from the professional tools at right; or let us help you choose from over 100 others.

Call one of our "tech reps" TODAY.

**Order before February 29, 1988 and mention "DD288" for these SPECIAL PRICES:**

| | List | Normal | SPECIAL |
|---|---|---|---|
| **Ctree** by Faircom | $395 | $315 | **$289** |
| **dBx Translator** | $350 | $299 | **$279** |
| **Essential Graphics** | $250 | $185 | **$165** |
| **PC Lint** | $139 | $119 | **$ 89** |
| **Windows for Data** | $295 | $239 | **$209** |

### 386 Development Tools

| | | |
|---|---|---|
| 386 Assembler/Linker | PC | $ 389 |
| 386 Debug - by Phar Lap | PC | $ 129 |
| 386/DOS Extender | PC | $ 919 |
| DESQview PS/2 | PC | $ 109 |
| F77L-EM - by Lahey | MS | Call |
| High C - by Metaware | PC | Call |
| OS/286 & 386 by AI Architects | PC | Call |

### AI Languages

| | | |
|---|---|---|
| APT - **A**ctive **P**rolog **T**utor - build applications interactively | PC | $ 49 |
| ARITY Prolog - full, 4 Meg | | |
| Interpreter - debug, C, ASM | PC | $ 229 |
| COMPILER/Interpreter-EXE | PC | $ 569 |
| Cogent Prolog Compiler | MS | $ 179 |
| MicroProlog Prof. Comp./Interp. | MS | $ 439 |
| PC Scheme LISP - by TI | PC | $ 85 |
| Star Sapphire | MS | $ 429 |
| TransLISP - learn fast | MS | $ 79 |
| TransLISP PLUS | MS | $ 149 |
| TURBO PROLOG by Borland | PC | $ 69 |

Others: IQ LISP ($239), IQC LISP ($269)

### Basic

| | | |
|---|---|---|
| BAS_C - economy | MS | $ 179 |
| BAS_PAS - economy | MS | $ 135 |
| Basic Development Tools | PC | $ 89 |
| db/Lib | MS | $ 119 |
| Exim Toolkit - full | PC | $ 45 |
| Finally - by Komputerwerks | PC | $ 85 |
| Inside Track | PC | $ 49 |
| Mach 2 by MicroHelp | PC | $ 55 |
| NetWorks by Exim | PC | $ 89 |
| QBase - screens | MS | $ 79 |
| QuickBASIC | PC | $ **69** |
| Quick Pak-by Crescent Software | PC | $ 59 |
| Quick-Tools by BC Associates | PC | $ 109 |
| Stay-Res | PC | $ 59 |
| True Basic | PC | $ 79 |
| Turbo BASIC - by Borland | PC | $ 69 |
| Turbo BASIC Database Toolbox | MS | $ 69 |

### FEATURES

**XQL** - SQL for Btrieve callable from BASIC, C, and Pascal or for interactive query. Computed fields, specify sort order, manipulate composite records from joined files. No royalties. MS $459

**Instant-C/16M** - Addresses up to 16M for program and data. Incremental compilation makes development faster than Turbo C (compile and relink XLISP in 4 secs vs 24). 286/386 only. PC, List: $895

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add $3/item.

### RECENT DISCOVERY

**SofTRAN,** the Translation and Text Language by TransOptima - full procedural language like C plus pattern and nonprocedural constructs cuts development effort by up to 16 times. PC $349

### C Language-Compilers

| | | |
|---|---|---|
| AZTEC C86 - Commercial | PC | $499 |
| C86 PLUS - by CI | MS | $359 |
| Datalight Optimum - C | MS | $ 99 |
| Lattice C - from Lattice | MS | $259 |
| Microsoft C 5.0- Codeview | MS | $**275** |
| Microsoft Quick C | MS | $ 67 |
| Rex - C/86 standalone ROM | MS | $695 |
| Turbo C by Borland | PC | $ 67 |

### C Libraries-Files

| | | |
|---|---|---|
| BTree by Soft Focus | MS | $ 69 |
| CBTREE - Source, no royalties | MS | $ 99 |
| ctree by Faircom - no royalties | MS | $315 |
| rtree - report generation | PC | $239 |
| dB2C Toolkit V2.0 | MS | $249 |
| dbQUERY - ad hoc, SQL-based | MS | Call |
| dbVISTA - Object only | MS | Call |
| Source - Single user | MS | Call |
| dBx - translator | MS | $299 |

### C-Screens, Windows, Graphics

| | | |
|---|---|---|
| C Worthy Interface Library | PC | $249 |
| Curses by Aspen Scientific | PC | $109 |
| dBASE Graphics for C | PC | $ 69 |
| ESSENTIAL GRAPHICS - fast | PC | $185 |
| FontWINDOW/PLUS | PC | $229 |
| GraphiC - new color version | PC | $279 |
| Greenleaf Data Windows w/source | PC | $155 |
| | PC | $259 |
| Terminal Mapping System | PC | $279 |
| TurboWINDOW/C - for Turbo C | PC | $ 75 |
| View Manager - by Blaise | PC | $199 |
| Windows for C - fast | PC | $149 |
| Windows for Data - validation | PC | $219 |
| Vitamin C - screen I/O | PC | $159 |
| VC Screen | PC | $ 79 |
| ZView - screen generator | MS | $149 |

### Atari ST & Amiga

We carry full lines of Manx & Lattice.

### DBASE Language

| | | |
|---|---|---|
| Clipper compiler | PC | $399 |
| dBASE II | MS | $329 |
| dBase III Plus | PC | $429 |

**Call for a catalog, literature, and *solid* value**

**800-421-8006**

**THE PROGRAMMER'S SHOP** ™

*Your complete source for software, services and answers*

**5-D Pond Park Road, Hingham, MA 02043**
**Mass: 800-442-8070 or 617-740-2510**

### RECENT DISCOVERY

**FORCE III,** Dbase Compiler by Sophco - small .EXEs, user-defined functions, I/O directives through BIOS/DOS/ANSI/FORCE/user-defined, extensions include FOR..NEXT loops, soundex, 1D arrays. Maverick. PC $109

### DBASE Language Cont.

| | | |
|---|---|---|
| dBASE III LANPack | PC | $649 |
| DBXL Interpreter by Word Tech | PC | $ 99 |
| FoxBASE+ Dev. - V2.0 | MS | $289 |
| Quicksilver by Word Tech | PC | $369 |

### DBASE Support

| | | |
|---|---|---|
| dAnalyst | PC | $ 89 |
| dBase Tools for C | PC | $ 65 |
| dBrief with Brief | PC | Call |
| dBC III by Lattice | MS | $169 |
| Documentor - dFlow superset | MS | $229 |
| Genifer by Bytel-code generator | MS | $279 |
| QuickCode III Plus | MS | $239 |
| R&R Report Writer | MS | $139 |
| Seek-It - Query-by-example | PC | $ 79 |
| Silver Comm Library | MS | $139 |
| Tom Rettig's Library | PC | $ 79 |
| UI Programmer - user interfaces | PC | $249 |

### DataBase & File Management

| | | |
|---|---|---|
| CQL | PC | $ 359 |
| DataFlex by Data Access | PC | $ 899 |
| DataFlex multiuser | PC | $1149 |
| Magic PC | PC | $ 699 |
| Paradox - original | PC | $ 369 |
| Paradox V2.0 | PC | $ 469 |
| Revelation by Cosmos | PC | $ 779 |

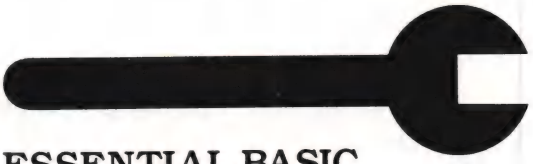
### Multilanguage Support

| | | |
|---|---|---|
| BTRIEVE ISAM | MS | $185 |
| BTRIEVE/N-multiuser | MS | $455 |
| GSS Graphics Dev't Toolkit | PC | $375 |
| HALO Development Package | MS | $389 |
| Graphics | PS | $209 |
| Help/Control - on line help | PC | $ 99 |
| Hoops Graphics Library | PC | $549 |
| Instant Programmer's Help | MS | $ 79 |
| Informix 4GL-application builder | PC | $839 |
| Informix SQL - ANSI standard | PC | $699 |
| NET-TOOLS - NET-BIOS | PC | $129 |
| Opt Tech Sort - sort, merge | MS | $ 99 |
| Norton Guides | PC | $ 75 |
| Panel Plus | MS | $395 |
| Pfinish - by Phoenix | MS | $229 |
| Report Option - for Xtrieve | MS | $109 |
| Screen Sculptor | PC | $ 89 |
| SSP/PC - 145+ math routines | PC | $269 |
| Synergy - create user interfaces | MS | $329 |
| Xtrieve - organize database | MS | $199 |
| ZAP Communications - VT 100 | PC | $ 89 |

# THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees and every product for Microcomputer Programming.

## EXIM TOOLKIT:



### ESSENTIAL BASIC PROGRAMMING TOOLS

A **must** for all BASIC programmers who use Microsoft's QuickBASIC or IBM's PC BASIC compilers, our feature-rich, easy-to-use **EXIM ToolKit** provides dozens of useful routines unavailable in BASIC or any other single software library sold today!

Here's just some of what the **EXIM ToolKit** can do:

With our **Screen Management** module, you get features like four-way scrolling, windowing and data entry/display in screen forms. You can also write to screen many times faster than with BASIC and develop software to copy areas of the screen for cut and paste functions.

The multi-user **Data Base Management** module lets multiple users access the same data files across a local area network and still maintain data integrity. Users can build, add and delete indexes; reformat files and much more. Now BASIC users can control data as never before.

The **Network Interface modules** make it possible for BASIC programs to issue NETBIOS commands.

BASIC limits you to 64K of RAM for strings. **EXIM ToolKit's Memory Manager** lets you access up to **256K additional RAM** for strings. And, we offer free technical support, 7 days a week.

Order your **EXIM ToolKit** today. It's only $99.95, ($199.95 with Network Interface Modules) plus $5.00 shipping and handling. (No licensing or royalties required.) If within 30 days, you are not completely satisfied with the **EXIM ToolKit,** return it to your dealer for a complete refund.

The **EXIM ToolKit** is so comprehensive, we're convinced it's the only BASIC Software Library you'll ever need.

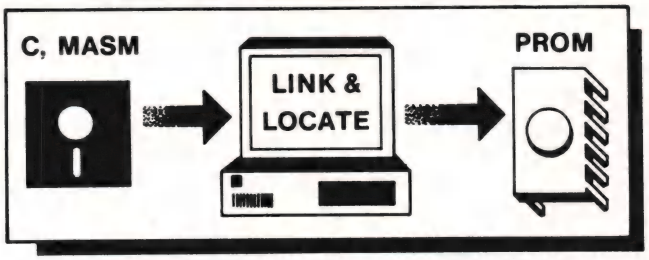
**PS: $ 45**

**EXIM**

**EXIM SERVICES OF N.A. INC.**
**P.O. Box 5417**
**Clinton, New Jersey 08809**
**(201) 735-7640**

CIRCLE NO. 143 ON READER SERVICE CARD

## FIRMWARE DEVELOPMENT



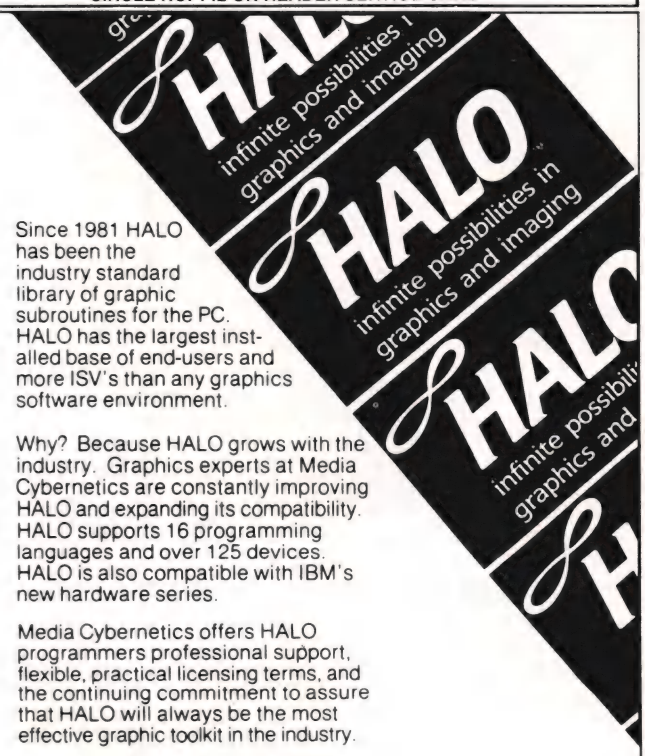

LINK & LOCATE enables PC users to produce ROM-based firmware for 8086/87/186 from object files generated by popular C compilers, such as from Microsoft, Lattice and Borland's Turbo C, and MASM from Microsoft, Provides full control of segment placement anywhere in memory. Supports output of Intel HEX file for PROM programmers, Intel OMF absolute object file for symbolic debuggers and in-circuit emulators. Includes Intel compatible linker, locator, librarian, hex formatter and cross reference generator. $350.

**NEW! Includes utility to support PC based PROM programmers.**

**SYSTEMS & SOFTWARE INC.** **PS: $329**

3303 Harbor Blvd., C11, Costa Mesa, CA 92626
Phone (714) 241-8650 FAX (714) 241-0377 TWX 910-695-0125

CIRCLE NO. 142 ON READER SERVICE CARD




Since 1981 HALO has been the industry standard library of graphic subroutines for the PC. HALO has the largest installed base of end-users and more ISV's than any graphics software environment.

Why? Because HALO grows with the industry. Graphics experts at Media Cybernetics are constantly improving HALO and expanding its compatibility. HALO supports 16 programming languages and over 125 devices. HALO is also compatible with IBM's new hardware series.

Media Cybernetics offers HALO programmers professional support, flexible, practical licensing terms, and the continuing commitment to assure that HALO will always be the most effective graphic toolkit in the industry.

**List: $300** **PS: $209**

CIRCLE NO. 144 ON READER SERVICE CARD

Call Today for FREE detailed information or try Risk-Free for 31 days, any product on this page.

HOURS: 8:30 A.M.-8:00 P.M. E.S.T.
5-D Pond Park Road, Hingham, MA 02043
Mass: 800-442-8070 or 617-740-2510

**800-421-8006**

**THE PROGRAMMER'S SHOP**
Your complete source for software services and answers

**Listing Two** *(Listing continued, text begins on page 30.)*

```
            3: ind3:=i;
         END;
         IF selector <> 1 THEN  {item not selected}
            attrib:=Low
         ELSE
         BEGIN
            attrib:=Rev;
            tind1:=ind1;
            tind2:=ind2;
            tind3:=ind3;
         END;
         WriteString(atree[ind1,ind2,ind3].title,
                     attrib);
         write('   '); {spaces to separate items}
      END;
      GoToXY(9,MenuLine4);
      clreol;
      write(atree[tind1,tind2,tind3].desc);

   END;


   PROCEDURE ProcessCr (VAR ind1,ind2,ind3,level,
                        selector,cmd_code:Byte);

   BEGIN
    {assign selector to index as it was picked}
    CASE level OF
       1: ind1:=selector;
       2: ind2:=selector;
       3: ind3:=selector;
    END;
    {if entry is terminal entry}
    IF atree[ind1,ind2,ind3].index = 0 THEN
    BEGIN
       {get associated cmd code}
       cmd_code:=atree[ind1,ind2,ind3].ccode;

       {process depending upon level}
       CASE level OF
          1: ind1:=0;
          2: BEGIN
                ind1:=0;
                ind2:=0;
                ind3:=0;
                level:=1;
             END;
          3: BEGIN
                ind2:=0;
                ind3:=0;
                level:=2;
             END;
       END;
    END
    ELSE
    BEGIN
       {down to next menu level}
       level:=level+1;
    END;
    {select set to 1st item}
    selector:=1;

   END;


   PROCEDURE ProcessMenu (VAR ind1,ind2,ind3,level,
                          selector,cmd_code:Byte);

   VAR

    key:          Char;
    position:     Integer;

   BEGIN

      key:=upcase(GetKey);
      CASE key OF
       rarrow: BEGIN
                  selector:=selector+1;
                  IF selector >
                     length(atree[ind1,ind2,ind3].chars) THEN
                      selector:=1;
               END;

       larrow: BEGIN
                  selector:=selector-1;
                  IF selector < 1 THEN
                     selector :=
                    length(atree[ind1,ind2,ind3].chars);
               END;

       endkey: selector :=
                 length(atree[ind1,ind2,ind3].chars);

         home: selector:=1;

           cr: ProcessCr(ind1,ind2,ind3,level,
                         selector,cmd_code);

          esc: cmd_code := 1;     {force exit}

   'A'..'Z','0'..'9'  {1st letter of menu item ?}
              : BEGIN
                   position :=
                     pos(key,atree[ind1,ind2,ind3].chars);
                   IF position <> 0 THEN
                   BEGIN              {is 1st letter}
                      selector:=position;
                      ProcessCr(ind1,ind2,ind3,level,
                                selector,cmd_code);
                   END
                   ELSE     {is not so beep}
                     Beep;
                END;
         ELSE               {invalid so beep}
            beep;
      END;

   END;


   {Procedure used to process the operator}
   {selected commands}
   {Defined in the main program's code}

   PROCEDURE ProcessCmd (cmd_code: Byte); FORWARD;


   FUNCTION DoMenu : Boolean;

   VAR

    Line: Integer;

   BEGIN

      {stop update of screen info}
      UpdateScreenStatus := False;

      {clear top 6 display lines}
      FOR Line := 1 TO 6 DO
      BEGIN
         GoToXY(1,Line);
         Clreol;
      END;

      {and 2nd to the last one}
      GoToXY(1,24);
      Clreol;

      DrawMenuFrame;
      {clear the menu exit flag}
      ExitMenu := False;

      {clear the program exit flag}
      ExitProgram := False;

      {selector to 1st item in menu}
      selector:=1;

      {command code initialized to 0 }
      cmd_code:=0;

      {1st tier of heirachy}
      level:=1;

      {root menu array location}
      ind1:=0;
      ind2:=0;
      ind3:=0;

      REPEAT

         DisplayMenu (ind1,ind2,ind3,level,selector);
         ProcessMenu (ind1,ind2,ind3,level,selector,
                      cmd_code);
         ProcessCmd  (cmd_code);
         cmd_code:=0;  {reset code selected last}

      UNTIL ExitMenu;

      {start update of screen info}
```

```
    UpdateScreenStatus := True;
    {return flag}
    DoMenu := ExitProgram;

END;


PROCEDURE Init_Menu;

{initialize the menu tree}

BEGIN

    fillchar(atree,sizeof(atree),0);

    WITH atree[0,0,0] DO
    BEGIN
       title:='Main Menu';
       chars:='QPDTFCE';
       index:=7;
    END;

    WITH atree[1,0,0] DO
    BEGIN
       title:='Quit';
       desc:='Exit Analyzer Menus';
       ccode:=1;
    END;

    WITH atree[2,0,0] DO
    BEGIN
       title:='Parameters';
       desc:='Set Serial Parameters';
       chars:='QBSWP';
       index:=5;
    END;

    WITH atree[2,1,0] DO
    BEGIN
       title:='Quit';
       desc:='Exit this submenu';
    END;

    WITH atree[2,2,0] DO
    BEGIN
       title:='Baud Rate';
       desc:='Change Serial Baud Rate';
       chars:='Q361249';
       index:=7;
    END;

    WITH atree[2,2,1] DO
    BEGIN
       title:='Quit';
       desc:='Exit this submenu';
    END;

    WITH atree[2,2,2] DO
    BEGIN
       title:='300';
       ccode:=2;
    END;

    WITH atree[2,2,3] DO
    BEGIN
       title:='600';
       ccode:=3;
    END;

    WITH atree[2,2,4] DO
    BEGIN
       title:='1200';
       ccode:=4;
    END;

    WITH atree[2,2,5] DO
    BEGIN
       title:='2400';
       ccode:=5;
    END;

    WITH atree[2,2,6] DO
    BEGIN
       title:='4800';
       ccode:=6;
    END;

    WITH atree[2,2,7] DO
    BEGIN
       title:='9600';
       ccode:=7;
END;

WITH atree[2,3,0] DO
BEGIN
   title:='Stop Bits';
   desc:='Set Number of Stop Bits';
   chars:='Q12';
   index:=3;
END;

WITH atree[2,3,1] DO
BEGIN
   title:='Quit';
   desc:='Exit this submenu';
END;

WITH atree[2,3,2] DO
BEGIN
   title:='1';
   ccode:=8;
END;

WITH atree[2,3,3] DO
BEGIN
   title:='2';
   ccode:=9;
END;

WITH atree[2,4,0] DO
BEGIN
   title:='Word Len.';
   desc:='Set Bits / Word';
   chars:='Q5678';
   index:=5;
END;

WITH atree[2,4,1] DO
BEGIN
  title:='Quit';
  desc:='Exit this SubMenu';
END;

WITH atree[2,4,2] DO
BEGIN
   title:='5';
   ccode:=10;
END;

WITH atree[2,4,3] DO
BEGIN
   title:='6';
   ccode:=11;
END;

WITH atree[2,4,4] DO
BEGIN
   title:='7';
   ccode:=12;
END;

WITH atree[2,4,5] DO
BEGIN
   title:='8';
   ccode:=13;
END;

WITH atree[2,5,0] DO
BEGIN
   title:='Parity';
   desc:='Set Serial Parity';
   chars:='QOEN';
   index:=4;
END;

WITH atree[2,5,1] DO
BEGIN
   title:='Quit';
   desc:='Exit this SubMenu';
END;

WITH atree[2,5,2] DO
BEGIN
   title:='Odd';
   ccode:=14;
END;

WITH atree[2,5,3] DO
BEGIN
   title:='Even';
   ccode:=15;
END;
```

*(continued on page 75)*

# Computer Faires show you how to coast into more business.


In 1988 the capitals of the computer coasts will be the sites of the most comprehensive events for more *new* business: The 13th West Coast Computer Faire and the 11th Northeast Computer Faire.

San Francisco and Boston are the hubs of regions ripe with people who know computers. Enthusiasts who have mastered the old and hunt for the new. Business people who know what to use and what to buy. People who will be coming to the shows to put their money where their knowledge is. And they'll be coming en masse; last year's West Coast Computer Faire drew in 43,500 attendees.

And we help you get the edge on bigger business than ever with highly-visible promotion, a highly-praised conference series, and highlights like State-of-the-Art Professional Development Seminars.

**Coast into Computer Faire business by reserving your exhibit space today. Call (617) 449-6600, ext. 4014.**

**THE 13TH WEST COAST**
**COMPUTER FAIRE**
**April 7–10, 1988 • Moscone Center • San Francisco, CA**

**THE 11TH NORTHEAST COMPUTER FAIRE**
**October 13-15, 1988 • World Trade Center • Boston, MA**

**I want to coast into more business!**

☐ Send me complete ☐ West Coast ☐ Northeast Computer Faire exhibitor information, including brochure, floorplan, and contracts.

☐ Send me attendee information.

☐ Have a Sales Representative call me today.

Name _____

Title _____

Company _____

Address _____

City _____ State_____ Zip_____

Telephone (_____) _____

Return to: Marketing Department
Computer Faire
300 First Avenue, Needham, MA 02194.

DRD 28

**Listing Two** *(Listing continued, text begins on page 30.)*

```
WITH atree[2,5,4] DO
BEGIN
   title:='None';
   ccode:=16;
END;

WITH atree[3,0,0] DO
BEGIN
  title:='Display';
  desc:='Select Channels for Display';
  chars:='Q12B';
  index:=4;
END;

WITH atree[3,1,0] DO
BEGIN
   title:='Quit';
   desc:='Exit this SubMenu';
END;

WITH atree[3,2,0] DO
BEGIN
   title:='COM1';
   ccode:=17;
END;

WITH atree[3,3,0] DO
BEGIN
   title:='COM2';
   ccode:=18;
END;

WITH atree[3,4,0] DO
BEGIN
   title:='Both 1 & 2';
   ccode:=19;
END;

WITH atree[4,0,0] DO
BEGIN
   title:='Trigger';
   desc:='Controls Trigger Modes';
   chars:='QCPMS';
   index:=5;
END;

WITH atree[4,1,0] DO
BEGIN
  title:='Quit';
  desc:='Exit this SubMenu';
END;

WITH atree[4,2,0] DO
BEGIN
   title:='Channel';
   desc:='Set Trigger Channel';
   chars:='Q12',
   index:=3;
END;

WITH atree[4,2,1] DO
BEGIN
   title:='Quit';
   desc:='Exit this SubMenu';
END;

WITH atree[4,2,2] DO
BEGIN
   title:='COM1';
   ccode:=20;
END;

WITH atree[4,2,3] DO
BEGIN
   title:='COM2';
   ccode:=21;
END;

WITH atree[4,3,0] DO
BEGIN
   title:='Pattern';
   desc:='Input Pattern for Trigger';
   ccode:=22;
END;

WITH atree[4,4,0] DO
BEGIN
   title:='Mode';
   desc:='Select Trigger Mode';
   chars:='QBAE';
   index:=4;
END;

WITH atree[4,4,1] DO
BEGIN
   title:='Quit';
   desc:='Exit this SubMenu';
END;

WITH atree[4,4,2] DO
BEGIN
   title:='Before';
   desc:='Display Data Before Trigger';
   ccode:=23;
END;

WITH atree[4,4,3] DO
BEGIN
   title:='After';
   desc:='Display Data After Trigger';
   ccode:=24;
END;

WITH atree[4,4,4] DO
BEGIN
   title:='Enable';
   desc:='Enable the Trigger';
   ccode:=25;
END;

WITH atree[4,5,0] DO
BEGIN
   title:='Stop';
   desc:='Stop waiting for Trigger Event';
   ccode:=26;
END;

WITH atree[5,0,0] DO
BEGIN
   title:='Format';
   desc:='Set Display Format';
   chars:='QAHS';
   index:=4;
END;

WITH atree[5,1,0] DO
BEGIN
   title:='Quit';
   desc:='Exit this SubMenu';
END;

WITH atree[5,2,0] DO
BEGIN
   title:='Ascii';
   desc:='Ascii Display Format';
   chars:='QNH';
   index:=3;
END;

WITH atree[5,2,1] DO
BEGIN
   title:='Quit';
   desc:='Exit this SubMenu';
END;

WITH atree[5,2,2] DO
BEGIN
   title:='Normal';
   desc:='Data Only';
   ccode:=27;
END;

WITH atree[5,2,3] DO
BEGIN
   title:='HandShake';
   desc:='Data and HandShake';
   ccode:=28;
END;

WITH atree[5,3,0] DO
BEGIN
   title:='Hex';
   desc:='Hex Display Format';
   chars:='QNH';
   index:=3;
END;

WITH atree[5,3,1] DO
BEGIN
   title:='Quit';
   desc:='Exit this SubMenu';
```

**Listing Two** *(Listing continued, text begins on page 30.)*

```
  END;

  WITH atree[5,3,2] DO
  BEGIN
     title:='Normal';
     desc:='Data Only';
     ccode:=29;
  END;

  WITH atree[5,3,3] DO
  BEGIN
     title:='HandShake';
     desc:='Data and HandShake';
     ccode:=30;
  END;

  WITH atree[5,4,0] DO
  BEGIN
     title:='Spaces';
     desc:='Insert space in data display';
     ccode:=31;
  END;

  WITH atree[6,0,0] DO
  BEGIN
     title:='Control';
     desc:='Alter Analyzer Operation';
     chars:='QASCR';
     index:=5;
  END;

  WITH atree[6,1,0] DO
  BEGIN
     title:='Quit';
     desc:='Exit this SubMenu';
  END;

  WITH atree[6,2,0] DO
  BEGIN
     title:='Acquire';
     desc:='Acquire Data';
     ccode:=32;
  END;

  WITH atree[6,3,0] DO
  BEGIN
     title:='Stop';
     desc:='Stop Data Acquisition';
     ccode:=33;
  END;

  WITH atree[6,4,0] DO
  BEGIN
     title:='Clear';
     desc:='Clear Screen of Data';
     ccode:=34;
  END;
  WITH atree[6,5,0] DO
  BEGIN
     title:='Reset';
     desc:='Reset Analyzer';
     ccode:=35;
  END;

  WITH atree[7,0,0] DO
  BEGIN
     title:='End';
     desc:='End the Analyzer Session';
     ccode:=36;
  END;
END;
```

**End Listing Two**

**Listing Three**

```
{*********************************************}
{***                                       ***}
{***          RS-232 support procedures    ***}
{***      for the serial protocol analyzer ***}
{***                written by             ***}
{***             Craig A. Lindley          ***}
{***                                       ***}
{***    Ver: 2.0     Last update: 08/15/87 ***}
{***                                       ***}
{*********************************************}

CONST

  COM1 = $3f8;              {com one port addr}
  COM2 = $2f8;              {com two port addr}
```

```
{table of values for the various baud rates}
{supported by the 8250. Rates from 50..9600}

BaudRate: ARRAY[1..15] OF Integer =
   ($900,$600,$417,$359,$300,$180,$0C0,$060,
    $040,$03A,$030,$020,$018,$010,$00C);

{8259 registers}

Int_Mask_Reg = $21; {interrupt enable register}
Int_Cmd_Reg  = $20; {command register}
End_Int_Cmd  = $20; {end of interrupt cmd}

{offsets from PortAddr for the various}
{8250 registers}

DLL             = 0;
DLM             = 1;
Int_Enable_Reg  = 1;
Int_Id_Reg      = 2;
LineControl     = 3;
ModemControl    = 4;
LineStatus      = 5;
ModemStatus     = 6;

{Status bit definitions}

DataRdyBit      = $01;
DTRBit          = $01;
Out2Bit         = $08;
ORBit           = $01;
RTSBit          = $02;
PEBit           = $02;
FEBit           = $04;
BrkBit          = $08;
CTSBit          = $10;
DSRBit          = $20;
TxRdyBit        = $20;
RIBit           = $40;
CDBit           = $80;

TYPE

 ParityType = (odd,even,none);

VAR

{Global storage of COM parameters}
{Used by the SetNewCOMParameter}
{procedure}

COM_Rate,
COM_StopBits,
COM_DataBits:  Integer;
COM_Parity:    ParityType;


{************** Serial Procedures ***************}

PROCEDURE Enable_Serial_Device (PortAddr:Integer);

VAR

 Temp:   Byte;

BEGIN

   {clear the 8250 serial device of any garbage}
   {by reading data port, int id reg. line}
   {status reg and modem status reg}

    Temp := port[PortAddr];
    Temp := port[PortAddr+Int_Id_Reg];
    Temp := port[PortAddr+LineStatus];
    Temp := port[PortAddr+ModemStatus];

    {read 8259 int mask reg and set IRQ3 or IRQ4}
    {low to enable requested interrupts.}
    {write result back to 8259 when finished}

    Temp := port[Int_Mask_Reg];
    IF PortAddr = COM1 THEN
       Temp := Temp AND $EF
    ELSE
       Temp := Temp AND $F7;
    port[Int_Mask_Reg] := Temp;

    {Out2, DTR and RTS high for 8250}
    port[PortAddr+ModemControl] :=
         Out2Bit + DTRBit + RTSBit;
```

*(continued on next page)*

**Listing Three** *(Listing continued, text begin on page 30.)*

```
    {all ints active except Tx}
    port[PortAddr+Int_Enable_Reg] := $0D;
    delay(100);

END;


PROCEDURE Disable_Serial_Devices;

VAR

 Temp:   Byte;

BEGIN

   {Read int mask in 8259, set both IRQ3}
   {and IRQ4 bits high to disable. Set Out2}
   {low for both COM1 and COM2 to prevent ints}
   {from leaving the async card and finally}
   {disable all interrupt sources in the UART}

   Temp := port[Int_Mask_Reg];
   Temp := Temp OR $18;
   port[Int_Mask_Reg] := Temp;
   port[COM1+ModemControl] := 0;
   port[COM2+ModemControl] := 0;
   port[COM1+Int_Enable_Reg] := 0;
   port[COM2+Int_Enable_Reg] := 0;

END;


PROCEDURE COM1_ISR;

{COM1 interrupt service routine}

VAR

 SerialInfo:  DataRec;

 Temp:        Integer;

 LineState,
 ModemState:  Byte;

BEGIN

   {read linestatus and modem status and}
   {combine into COM1_Status. See text for}
   {bit encoding.}

   LineState := port[COM1+LineStatus];
   ModemState:= port[COM1+ModemStatus];
   Temp := (LineState SHR 1) AND $0F;

   {COM1_Status has the UART state in 8 bits}
   {this is comprised of status info only, no data}

   COM1_Status := (ModemState AND $F0) OR lo(Temp);

   {if there is data to receive and we are}
   {acquiring data}

   IF (((LineState AND DataRdyBit) <> 0) AND
       COM1_Data_Acquire) THEN
   BEGIN
      {if there is room in the COM1 input fifo}
      IF COM1_Input_Fifo.Ovd.Count <
                  SerialDataFifoSize THEN
      BEGIN
         {Put received data and status into fifo}
         SerialInfo.Data   := port[COM1];
         SerialInfo.Status := COM1_Status;
         PutSerialData(SerialInfo,COM1_Input_Fifo);
      END
      ELSE
      BEGIN
         writeln('COM1 Input fifo overflowed');
         halt;
     END;
   END;

   {signal end of int to 8259}
   port[Int_Cmd_Reg] := End_Int_Cmd;

END;
```

```
PROCEDURE COM2_ISR;

{COM2 interrupt service routine}

VAR

 SerialInfo:  DataRec;

 Temp:        Integer;

 LineState,
 ModemState:  Byte;

BEGIN

   {read linestatus and modem status and}
   {combine into COM2_Status. See text for}
   {bit encoding.}

   LineState := port[COM2+LineStatus];
   ModemState:= port[COM2+ModemStatus];
   Temp := (LineState SHR 1) AND $0F;

   {COM2_Status has the UART state in 8 bits}
   {this is comprised of status info only, no data}

   COM2_Status := (ModemState AND $F0) OR lo(Temp);

   {if there is data to receive and we are}
   {acquiring data}

   IF (((LineState AND DataRdyBit) <> 0) AND
       COM2_Data_Acquire) THEN
   BEGIN
      {if there is room in the COM2 input fifo}
      IF COM2_Input_Fifo.Ovd.Count <
                  SerialDataFifoSize THEN
      BEGIN
        {Put received data and status into fifo}
        SerialInfo.Data   := port[COM2];
        SerialInfo.Status := COM2_Status;
        PutSerialData(SerialInfo,COM2_Input_Fifo);
      END
      ELSE
      BEGIN
         writeln('COM2 Input fifo overflowed');
         halt;
      END;
   END;

   {signal end of int to 8259}
   port[Int_Cmd_Reg] := End_Int_Cmd;

END;


PROCEDURE COM1_Int_Service_Routine;

BEGIN

 INLINE($50/$53/$51/$52/$57/    {Push ax,bx,cx,dx,}
        $56/$06/$1e/            {di,si,es,ds}
        $2e/$a1/turbodseg/      {mov ax,cs:turbodseg}
        $8e/$d8/                {mov ds,ax}
        $fb);                   {sti}

 COM1_ISR;

 {standard interrupt service routine postamble}

 INLINE($fa/$1f/$07/$5e/$5f/    {interrupts off}
        $5a/$59/$5b/$58/        {Pop ds,es,si,di,}
                                {dx,cx,bx,ax}
        $5d/$5d/$cf);           {trash sp, restore}
                                {Bp and iret}
 END;


 PROCEDURE COM2_Int_Service_Routine;

 BEGIN

  INLINE($50/$53/$51/$52/$57/   {Push ax,bx,cx,dx,}
         $56/$06/$1e/           {di,si,es,ds}
         $2e/$a1/turbodseg/     {mov ax,cs:turbodseg}
         $8e/$d8/               {mov ds,ax}
         $fb);                  {sti}

  COM2_ISR;

  {standard interrupt service routine postamble}
```

*(continued on next page)*

**Listing Three** *(Listing continued, text begins on page 30.)*

```
  INLINE($fa/$1f/$07/$5e/$5f/    {interrupts off}
         $5a/$59/$5b/$58/        {Pop ds,es,si,di,}
                                 {dx,cx,bx,ax}
         $5d/$5d/$cf);           {trash sp, restore}
                                 {Bp and iret}

END;


PROCEDURE Install_Serial_Handlers;

BEGIN

 WITH regs DO       {install into IRQ3 & 4}
 BEGIN
    ah := $35;          {get vector func. code}
    al := $0B;          {for IRQ3}
    msdos(regs);        {call dos to get vector}
    OldIRQ3_CS := es;{save code seg}
    OldIRQ3_IP := bx;{and instruction ptr}

    ah := $35;          {get vector func. code}
    al := $0C;          {for IRQ4}
    msdos(regs);        {call dos to get vector}
    OldIRQ4_CS := es;{save code seg}
    OldIRQ4_IP := bx;{and instruction ptr}

    ah := $25;          {set vector func. code}
    al := $0C;          {for IRQ4}
    ds := cseg;         {code in our segment}
                        {at this offset}
    dx := ofs(COM1_Int_Service_Routine);
    msdos(regs);        {call dos to set vector}

    ah := $25;          {set vector func. code}
    al := $0B;          {for IRQ3}
    ds := cseg;         {code in our segment}
                        {at this offset}
    dx := ofs(COM2_Int_Service_Routine);
    msdos(regs);        {call dos to set vector}

 END;
END;


PROCEDURE Remove_Serial_Handlers;

BEGIN
   {Put saved vectors for IRQ3 and IRQ4 back}
   WITH regs DO
   BEGIN
      ah := $25;
      al := $0C;
      ds := OldIRQ4_CS;
      dx := OldIRQ4_IP;
      msdos(regs);

      ah := $25;
      al := $0B;
      ds := OldIRQ3_CS;
      dx := OldIRQ3_IP;
      msdos(regs);
   END;

END;


PROCEDURE Set_Serial_Parameters
     (PortAddr,Baud,StopBits,DataBits:Integer;
      Parity: ParityType);
VAR

 Temp,
 Rate:   Integer;

BEGIN

   {set DLAB high for divisor regs}
   port[PortAddr+LineControl] := $80;

   {look up rate word in table}
   Rate := BaudRate[Baud];

   {MSB into most sign divisor reg}
   port[PortAddr+DLM] := hi(Rate);

   {LSB into less sign divisor reg}
   port[PortAddr+DLL] := lo(Rate);

   {move databits into 2 least sign bits}
   {add in stop bit into bit pos 2}
   {set parity enable and parity even}
   {bits if appropriate}

   Temp := (DataBits - 5) AND $03;
   Temp := Temp OR ((StopBits - 1) SHL 2);
   CASE Parity OF
     odd:  Temp := Temp + $08;
    even: Temp := Temp + $18;
    none: ;
   END;

   {remove DLAB and setup parameters}
   port[PortAddr+LineControl] := lo(Temp);
   delay(100);

END;


FUNCTION Get_Serial_Status (PortAddr:Integer)
            :Integer;

VAR

 Temp:  Integer;

BEGIN

   {Get full 16 bits of COM port status}
   {grouped as ModemStatus:LineStatus}

   Temp := port[PortAddr+ModemStatus];
   Temp := Temp SHL 8;
   Temp := Temp OR port[PortAddr+LineStatus];
   Get_Serial_Status := Temp;

END;


PROCEDURE SetNewCOMParameter;

BEGIN

   {take the COM ports down}
   Disable_Serial_Devices;

   {Set the COM ports to the global parameters}

   Set_Serial_Parameters(COM1,COM_Rate,
          COM_StopBits,COM_DataBits,COM_Parity);
   Set_Serial_Parameters(COM2,COM_Rate,
          COM_StopBits,COM_DataBits,COM_Parity);

   {bring COM ports back up}
   Enable_Serial_Device(COM1);
   Enable_Serial_Device(COM2);

END;
```

**End Listing Three**

## Listing Four

```
{$K-}              {Compiler switch - never change}

{***********************************************}
{***                                         ***}
{***                Turbo Pascal             ***}
{***            Multitasking Kernel          ***}
{***                 written by              ***}
{***              Craig A. Lindley           ***}
{***                                         ***}
{***    Ver: 2.0     Last update: 08/15/87   ***}
{***                                         ***}
{***********************************************}

CONST

   task_stack_size  = 1000;
```

```
PROCEDURE Yield;

{This version in assembly language for}
{speed. See version above for comments.}

BEGIN

   IF cp^.link <> cp THEN  {must have more than}
                           {one task forked to be}
                           {able to yield}
   BEGIN
      INLINE($C6/$06/child_process/$00/
                              {child_process is false}
             $C4/$3E/cp/      {les di,[cp]}
             $89/$E0/         {mov ax,sp}
             $05/$02/$00/     {add ax,2}
             $26/             {es:}
             $89/$45/$04/     {mov [di+4],ax}
             $26/             {es:}
             $C6/$45/$06/$00/ {mov byte ptr [di+6],0}
             $89/$FB/         {L1: mov bx,di}
             $26/             {es:}
             $C4/$1F/         {les bx,[bx]}
             $26/             {es:}
             $80/$7F/$06/$00/ {cmp byte ptr [bx+6],0}
             $74/$04/         {je L2}
             $89/$DF/         {mov di,bx}
             $EB/$F0/         {jmp L1}
             $89/$1E/cp/      {L2: mov [cp],bx}
             $8C/$06/cp+2/    {    mov [cp+2],es}
             $26/             {es:}
             $C6/$47/$06/$02/ {mov byte ptr [bx+6],2}
             $26/             {es:}
             $8B/$6F/$04);    {mov bp,[bx+4]}
   END
   ELSE
   BEGIN
      writeln('Cannot yield only single task running');
      halt;
   END;

END;
PROCEDURE Wait;  {put current task in wait mode}
                 {until a send makes it ready}

BEGIN

   IF cp^.link <> cp THEN  {must have more than}
                           {one task forked to be}
                           {able to wait}

   BEGIN
      waitfor^ := cp;      {waitfor points at the}
                           {current task}
      INLINE($C6/$06/child_process/$00/
                              {child_process is false}
             $C4/$3E/cp/      {les di,[cp]}
             $89/$E0/         {mov ax,sp}
             $05/$02/$00/     {add ax,2}
             $26/             {es:}
             $89/$45/$04/     {mov [di+4],ax}
             $26/             {es:}
             $C6/$45/$06/$01/ {mov byte ptr [di+6],1}
             $89/$FB/         {L1: mov bx,di}
             $26/             {es:}
             $C4/$1F/         {les bx,[bx]}
             $26/             {es:}
             $80/$7F/$06/$00/ {cmp byte ptr [bx+6],0}
             $74/$04/         {je L2}
             $89/$DF/         {mov di,bx}
             $EB/$F0/         {jmp L1}
             $89/$1E/cp/      {L2: mov [cp],bx}
             $8C/$06/cp+2/    {    mov [cp+2],es}
             $26/             {es:}
             $C6/$47/$06/$02/ {mov byte ptr [bx+6],2}
             $26/             {es:}
             $8B/$6F/$04);    {mov bp,[bx+4]}
   END
   ELSE
   BEGIN
      writeln('Cannot wait only single task running');
      halt;
   END;

END;
```

**End Listings**

**Listing One** *(Text begins on page 92.)*

```
        Listing 1 -- fdump.c, Printed 10/29/1987
  1| #include <stdio.h>
  2| #include <fcntl.h>
  3| #include <ctype.h>
  4|
  5| /*  Binary file dump utility. Usage is:
  6|  *    fdump file...    Dump all files listed on command line
  7|  *    fdump +N file    Start N bytes from start of file
  8|  *    fdump -N file    Start N bytes from end of file.
  9|  */
 10|
 11| extern long     lseek( int, long, int );
 12|
 13| /*------------------------------------------------------------*/
 14|
 15| #define BUFSIZE 16
 16| #define CR      0x0d
 17| #define LF      0x0a
 18| #define C_CR    0x11    /* Print left arrow for CR      */
 19| #define C_LF    0x19    /* Print down arrow for LF      */
 20|
 21| #define isprinting(c) (' ' <= (c)  &&  (c) <= '~')
 22|
 23| static   long   Bytenum = 0L;
 24|
 25| /*------------------------------------------------------------*/
 26|
 27| main(argc, argv)
 28| int     argc;
 29| char    **argv;
 30| {
 31|     register int fd;
 32|     long         offset = 0L;
 33|
 34|     if( argc == 1 )
 35|         usage();
 36|
 37|     ++argv;
 38|     --argc;
 39|
 40|     if( **argv == '-' || **argv == '+' )
 41|     {
 42|         if( !isdigit( argv[0][1] ) )
 43|             usage();
 44|
 45|         offset = atoi( argv[0] );
 46|
 47|         ++argv;
 48|         --argc;
 49|     }
 50|
 51|     for(; --argc >= 0 ; ++argv )
 52|     {
 53|         if( (fd = open( *argv, O_RDONLY | O_BINARY)) == -1 )
 54|             perror( *argv );
 55|         else
 56|         {
 57|             if( offset > 0 )
 58|                     Bytenum = lseek( fd, offset, SEEK_SET );
 59|
 60|             else if( offset < 0 )
 61|                     Bytenum = lseek( fd, offset, SEEK_END );
 62|
 63|             dofile( fd );
 64|             close ( fd );
 65|         }
 66|     }
 67|
 68|     exit(0);
 69| }
 70|
 71| /*------------------------------------------------------------*/
 72|
 73| usage()
 74| {
 75|     fprintf(stderr, "Usage fdump [+|- N] file...\n");
 76|     fprintf(stderr, "+N   Start dump at byte N\n");
 77|     fprintf(stderr, "-N   Start dump N bytes from EOF\n");
 78|     exit( 1 );
 79| }
 80|
 81| /*------------------------------------------------------------*/
 82|
 83| dofile( fd )
 84| int     fd;
 85| {
 86|         static   char   buf[ BUFSIZE ];
 87|         register char   nbytes;
 88|
 89|         while( (nbytes = read(fd, buf, BUFSIZE)) > 0 )
 90|                 doline( buf, nbytes );
 91| }
 92|
 93| /*------------------------------------------------------------*/
 94|
 95| doline( buf, nbytes )
 96| char    *buf;
 97| {
 98|     static   char       hex[] = "0123456789abcdef" ;
 99|     register int        i;
100|     register char       *p;
101|
102|     printf("%06lx: ", Bytenum );
103|     Bytenum += nbytes;
104|
105|     for( i = 1, p = buf ; i <= BUFSIZE ; i++ , p++ )
106|     {
107|             if( i <= nbytes )
108|             {
109|                     putchar( hex[ (*p >> 4) & 0xf ] );
110|                     putchar( hex[  *p       & 0xf ] );
111|             }
112|             else
113|             {
114|                     putchar( ' ' );
115|                     putchar( ' ' );
116|             }
117|
118|             putchar( ' ' );
119|     }
120|
121|     putchar(' ');
122|
123|     for( i = 1, p = buf ; i <= nbytes ; i++ , p++ )
124|     {
125|             if( *p == CR )
126|                     putchar( C_CR );
127|
128|             else if( *p == LF )
129|                     putchar( C_LF );
130|             else
131|                     putchar( isprinting(*p) ? *p : '.' );
132|     }
133|
134|     putchar('\n');
135|     putchar('\r');
136| }
```

**End Listing**

# UNLEASH YOUR 80386!

Your 80386-based PC should run two to three times as fast as your old AT. This speed-up is primarily due to the doubling of the clock speed from 8 to 16 MHz. The new MicroWay products discussed below take advantage of the real power of your 80386, which is actually 4 to 16 times that of the old AT! These new products take advantage of the 32 bit registers and data bus of the 80386 and the Weitek 1167 numeric coprocessor chip set. They include a family of MicroWay 80386 compilers that run in protected mode and numeric coprocessor cards that utilize the Weitek technology.

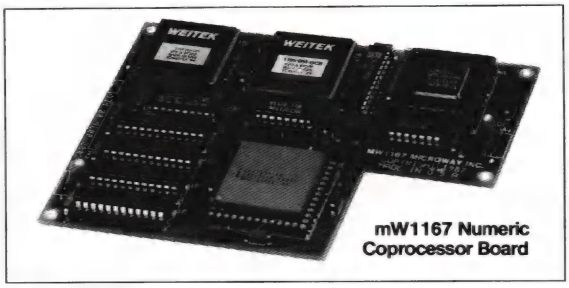
The benefits of our new technologies include:

- An increase in addressable memory from 640K to 4 gigabytes using MS-DOS or Unix.
- A 12 fold increase in the speed of 32 bit integer arithmetic.
- A 4 to 16 fold increase in floating point speed over the 80387/80287 numeric coprocessors.

Equally important, whichever MicroWay product you choose, you can be assured of the same excellent pre- and post-sales support that has made MicroWay the world leader in PC numerics and high performance PC upgrades. For more information, please call the Technical Support Department at

**617-746-7341**

After July 1988 call 508-746-7341



mW1167 Numeric Coprocessor Board

# MicroWay® 80386 Support

## MicroWay 80386 Compilers

**NDP Fortran-386** and **NDP C-386** are globally optimizing 80386 native code compilers that support a number of Numeric Data Processors, including the 80287, 80387 and mW1167. They generate mainframe quality optimized code and are syntactically and operationally compatible to the Berkeley 4.2 Unix f77 and PCC compilers. MS-DOS specific extensions have been added where necessary to make it easy to port programs written with Microsoft C or Fortran and R/M Fortran.

The compilers are presently available in two formats: Microport Unix 5.3 or MS-DOS as extended by the Phar Lap Tools. MicroWay will port them to other 80386 operating systems such as OS/2 as the need arises and as 80386 versions become available.

The key to addressing more than 640 kbytes is the use of 32-bit integers to address arrays. NDP Fortran-386 generates 32-bit code which executes 3 to 8 times faster than the current generation of 16-bit compilers. There are three elements each of which contributes a factor of 2 to this speed increase: very efficient use of 80386 registers to store 32-bit entities, the use of inline 32-bit arithmetic instead of library calls, and a doubling in the effective utilization of the system data bus.

An example of the benefit of excellent code is a 32-bit matrix multiply. In this benchmark an NDP Fortran-386 program is run against the same program compiled with a 16-bit Fortran. Both programs were run on the same 80386 system. However, the 32-bit code ran 7.5 times faster than the 16-bit code, and 58.5 times faster than the 16-bit code executing on an IBM PC.

**NDP FORTRAN-386™ .....$595**
**NDP C-386™ .....$595**

## MicroWay Numerics

The **mW1167™** is a MicroWay designed high speed numeric coprocessor that works with the 80386. It plugs into a 121 pin "Weitek" socket that is actually a super set of the 80387. This socket is available on a number of motherboards and accelerators including the AT&T 6386, Tandy 4000, Compaq 386/20, Hewlett Packard RS/20 and MicroWay Number Smasher 386. It combines the 64-bit Weitek 1163/64 floating point multiplier/adder with a Weitek/Intel designed "glue chip". The mW1167™ runs at 3.6 MegaWhetstones (compiled with NDP Fortran-386) which is a factor of 16 faster than an AT and 2 to 4 times faster than an 80387.

**mW1167 16 MHz .....$1495**
**mW1167 20 MHz .....$1995**

**Monoputer™** – The INMOS T800-20 Transputer is a 32-bit computer on a chip that features a built-in floating point coprocessor. The T800 can be used to build arbitrarily large parallel processing machines. The Monoputer comes with either the 20 MHz T800 or the T414 (a T800 without the NDP) and includes 2 megabytes of processor memory. Transputer language support from MicroWay includes Occam, C, Fortran, Pascal and Prolog.

**Monoputer T414-20 with 2 meg[1] ...$1495**
**Monoputer T800-20 with 2 meg[1] ...$1995**

**Quadputer™** can be purchased with 2, 3 or 4 transputers each of which has 1 or 4 megabytes of memory. Quadputers can be cabled together to build arbitrarily fast parallel processing systems that are as fast or faster than today's mainframes. A single T800 is as fast as an 80386/mW1167 combination!

**Biputer™ T800/T414 with 2 meg[1] ....$3495**
**Quadputer 4 T414-20 with 4 meg[1] ...$6000**

[1]Includes Occam

## 80386 Multi-User Solutions

**AT8™** – This intelligent serial controller series is designed to handle 4 to 16 users in a Xenix or Unix environment with as little as 3% degradation in speed. It has been tested and approved by Compaq, Intel, NCR, Zenith, and the Department of Defense for use in high performance 80286 and 80386 Xenix or Unix based multi-user systems.

**AT4 – 4 users .....$795**
**AT8 – 8 users .....$995**
**AT16 – 16 users .....$1295**

**Phar Lap™** created the first tools that make it possible to develop 80386 applications which run under MS-DOS yet take advantage of the full power of the 80386. These include an 80386 monitor/loader that runs the 80386 in protected linear address mode, an assembler, linker and debugger. These tools are required for the MS-DOS version of the MicroWay NDP Compilers.

**Phar Lap Tools .....$495**

## PC/AT ACCELERATORS

**287Turbo-10 10 MHz .....$450**
**287Turbo-12 12 MHz .....$550**
**287TurboPlus-12 12 MHz .....$629**
**FASTCACHE-286 9 MHz .....$299**
**FASTCACHE-286 12 MHz .....$399**
**SUPERCACHE-286 .....$499**

## MATH COPROCESSORS

**80387-20 20 MHz .....$895**
**80387-16 16 MHz .....$495**
**80287-10 10 MHz .....$349**
**80287-8 8 MHz .....$259**
**80287-6 6 MHz .....$179**
**8087-2 8 MHz .....$154**
**8087 5 MHz .....$99**

MicroWay

*The World Leader in PC Numerics*

*P.O. Box 79, Kingston, Mass. 02364 USA (617) 746-7341*
*32 High St., Kingston-Upon-Thames, U.K., 01-541-5466*
*St. Leonards, NSW, Australia 02-439-8400*

**Listing One** *(Text begins on page 100.)*

```
Unit criterr;

 { Critical error handler, Turbo Pascal Release 4.0 }

Interface
Uses dos, crt;

{ EXTERNALLY VISIBLE PORTION }

   { The following are for saving and restoring the screen,     }
   { which is assumed to be in text mode and display page 0     }

Const bell = #7;

Type scrnPtr = ^scrnBuffer;
     scrnBuffer = array [1..4096] of byte;

Var  display, saveNode  : scrnPtr;              { display buffer }

   { The following are global variables available to the using  }
   { program to find out if an error occurred and, if so, what  }
   { it was. The program can then take appropriate action.      }

     criticalErrorOccurred : boolean;
     criticalErrorCode     : integer;
     criticalErrorDrive    : integer;
     criticalActionCode    : char;

   { The only externally visible routine installs the critical  }
   { error handler in Int 24h, replacing the DOS default.       }

Procedure InstallCEH;
Implementation
{ ------------------------------------------------------------ }

   { Following is a general-purpose critical error handler }

{$F+}
Procedure CEHandler (
      Flags, CS, IP, AX, BX, CX, DX, SI, DI, DS, ES, BP : word);
Interrupt;

Var  AH, AL    : byte;
     row, col  : integer;
     action    : char;
{ -------------------------- }
  { Local functions }

  { giveReason lists reason for critical error by decoding the  }
  { low byte of the DI register. Called by procs DiskError and  }
  { CharDeviceError. Writes to screen.                          }

  Procedure GiveReason (error : byte);
  Begin
    Case error of
      $00: Writeln ('Write protect');
      $01: Writeln ('Unknown unit');
      $02: Writeln ('Drive not ready');
      $03: Writeln ('Unknown command');
      $04: Writeln ('CRC data error');
      $05: Writeln ('Bad request structure length');
      $06: Writeln ('Seek error');
      $07: Writeln ('Unknown media type');
      $08: Writeln ('Sector not found');
      $0A: Writeln ('Write fault');
      $0B: Writeln ('Read fault');
      $0C: Writeln ('General failure');
      $0D: Writeln ('Bad file allocation table');
      else Writeln ('Unknown');
    End;
  End;
{ -------------------------- }

  { DiskError is dispatched when H/O bit of AH is 0 }

  Function DiskError : word;

  Var   area, why : byte;

  Begin
    Writeln;
    CriticalErrorDrive := AL;
    Writeln ('Disk error on drive ', char (AL + 65));
    Area := (AH and 6) shr 1;                  { get AH bits 1-2 }
    Case area of
      0: Writeln ('Error in DOS communications area');
      2: Writeln ('Error in disk directory');
      3: Writeln ('Error in files area');
    End;
    Why := lo (DI);
    Write ('Type of error: ');
    GiveReason (why);
    DiskError := why;                          { error return code }
  End;
{ -------------------------- }

  { NonDiskError is dispatched when H/O bit of AH is 1. }
  { Usually triggered by a printer problem or bad FAT.  }

  Function NonDiskError : word;

  Var  why         : byte;
       deviceAttr  : ^word;
       deviceName  : ^char;
       ch          : shortInt;

  Begin
    DeviceAttr := ptr (BP, SI+4);   { point to device attr word }
    If (deviceAttr^ and $8000) <> 0 then    { if bit 15 is on.. }
      Begin
        Writeln ('Character device error');
        Write ('Failing device is ');
        ch := 0;
        Repeat
          deviceName := ptr (BP, SI + $0A + ch);
          Write (deviceName^);
          inc (ch);
        Until (deviceName^ = chr (0)) or (ch > 7);
        Writeln;
      End
    Else                                       { assume bad FAT }
      Begin
        Writeln ('Disk error has occurred');
        Write   ('Probable cause: ');
        Why := $0D;
        GiveReason (why);
      End;
    NonDiskError := why;                       { return error code }
  End;
{ -------------------------- }

Begin   { Body of CEHandler procedure }
  CriticalErrorOccurred := TRUE;               { set global flag }
  AH := hi (AX);
  AL := lo (AX);
  Col := whereX;                  { get current cursor position }
  Row := whereY;
  New (saveNode);
```

```
  SaveNode^ := display^;                    { and save screen image }
  Write (bell);                               { beep to alert user }
  If (AH and $80) = 0 then                      { if AH bit 7 = 0 }
    CriticalErrorCode := DiskError
  Else
    CriticalErrorCode := NonDiskError;
  Repeat                    { what are we gonna do about the error? }
    Write ('Abort/Retry/Ignore? ');
    Action := upCase (readKey);
    Writeln (action);
  Until action in ['A', 'I', 'R'];
  CriticalActionCode := action;
  If action = 'I' then begin  { pretend the error didn't happen }
    CriticalErrorOccurred := FALSE;
    CriticalErrorCode     := 0;
    CriticalErrorDrive    := $FF;
    CriticalActionCode    := ' ';
  End;
  Display^ := saveNode^;                    { restore screen image }
  Dispose (saveNode);
  Gotoxy (col, row);                      { restore cursor position }
  AX := 0;                             { tell DOS to ignore the error }
End;
{$F-}
{ ------------------------------------------------------------------ }

   { Externally visible: installs the error handler.                 }
   { NOTE: Program termination automatically reinstalls the          }
   { default handler in the vector table.                            }

Procedure InstallCEH;

Var   videoMode : byte absolute $0040 : $0049;

Begin
  SetIntVec ($24, @CEHandler);                { install in int 24h }
  CriticalErrorOccurred := FALSE;                    { set globals }
  CriticalErrorCode     := 0;
  CriticalErrorDrive    := $FF;
  CriticalActionCode    := ' ';
  If videoMode = 7 then
    Display := ptr ($B000, $0000)             { set display address }
  Else
    Display := ptr ($B800, $0000);
End;

End.
```

**End Listing One**

## Listing Two

```
Program cerrtest;

    { Test critical error handler }

Uses  crt, dos, criterr;

Var testFile   : text;
    n, ignored : integer;

Begin
  {$I-}
  ClrScr;
  InstallCEH;
  For n := 1 to 10 do
    Writeln ('This is output line ', n);
  Assign (testFile, 'A:TEST.FIL');

  Repeat
    Rewrite (testFile);
    Ignored := IOResult;              { clear system error status }
  Until criticalActionCode <> 'R';

  Writeln ('After disk attempt, criticalErrorOccurred = ',
           criticalErrorOccurred);
  Writeln (' and criticalErrorCode = ', criticalErrorCode);
  Writeln (' and criticalErrorDrive = ', criticalErrorDrive);
  Writeln (' and criticalActionCode = ', criticalActionCode);
  {$I+}
End.
```

**End Listings**

# THE FORTH COLUMN

## Listing One *(Text begins on page 108.)*

```
****** Listing 1 for TRACY, Feb '88 ******

( Stream data and text read and write )
These utilities read and write streams of data and text from
standard or BLOCKed files.

Text lines are read into the user buffer until either the buffer
is full, or the file is empty, or an #EOF or #EOL is read.
The terminating #EOF or #EOL , if present, is not read into the
buffer.  #LF ( linefeeds) are read but ignored.

Output files are assumed to be extensible.

For your convenience, the Standard Prelude and DDJ Controlled
Reference Word Set are duplicated in this file.



( LOAD screen for DDJ Standard Prelude and String Extension)
( MJT  Nov 22 1987 for DDJ February 1987)

(     2 LOAD ( Standard prelude)
      3 LOAD ( Augmented interpretation)
   4  5 THRU ( Controlled words)
   6  9 THRU ( Strings)
  10 13 THRU ( General file support)
\    14 LOAD ( Read and write data files)
  15 16 THRU ( Read and write BLOCKed data files)
     17 LOAD ( Read text file, no   #EOL)
\    18 LOAD ( Read text file, with #EOL)
     19 LOAD ( Write text file)
  20 22 THRU ( Some examples)


( FORTH-83 functions-- typical definitions)
( Adjust these words for your Forth.    See DDJ Oct 1987.)
( Note: functions already provided need not be redefined.)
: RECURSE   [COMPILE] MYSELF ;  IMMEDIATE
: INTERPRET   INTERPRET ;

: I> ( - 'data)   COMPILE R> ;  IMMEDIATE
: >I ( - 'data)   COMPILE >R ;  IMMEDIATE

( Used for alignment: )
: ALIGN     ( HERE 1 AND ALLOT) ;
: REALIGN   ( a - a' )   ( DUP 1 AND +) ;

2 CONSTANT CELL    : CELL+   2+ ;     : CELLS   2* ;

: UNDO    I> R> R> 2DROP >I ;  \ Undoes a DO-- LOOP.
( Required definitions - used to support further compilation)

: THRU ( n n2)   1+ SWAP DO  I LOAD  LOOP ;
\ LOADS screens n through n2.

: \   >IN @ 64 + -64 AND >IN ! ;  IMMEDIATE
\ comment to end of line.  For use in screens only.

: \\   1024 >IN ! ;  IMMEDIATE
\ stops interpreting or compiling screen immediately.

: \IF ( f )   0= IF  [COMPILE] \  THEN ;  IMMEDIATE
\ conditional interpretation or compilation.

: NEED ( - f)   32 ( ie blank) WORD FIND  SWAP DROP  0= ;
\ true if the following word is in the search order.
\ FORTH-83 Controlled Words

NEED  2* \IF   :  2*    DUP   + ;
NEED D2* \IF   : D2*   2DUP D+ ;

NEED HEX \IF   : HEX   16 BASE ! ;
NEED  C, \IF   : C, ( n )    HERE 1 ALLOT C! ;

NEED BL \IF  32 CONSTANT BL

NEED ERASE \IF   : ERASE ( a n)    00 FILL ;
NEED BLANK \IF   : BLANK ( a n)    BL FILL ;

NEED .R \IF   : .R ( n width)    >R DUP 0< R> D.R ;


\ DDJ Forth Column Controlled Words
NEED 2>R
\IF : 2>R    COMPILE SWAP COMPILE >R COMPILE >R ;  IMMEDIATE
NEED 2R>
\IF : 2R>    COMPILE R> COMPILE R> COMPILE SWAP ;  IMMEDIATE
NEED @EXECUTE \IF   : @EXECUTE    @ EXECUTE ;

NEED AGAIN
\IF   : AGAIN    0 [COMPILE] LITERAL [COMPILE] UNTIL ;  IMMEDIATE
NEED DLITERAL
DUP \IF   : DLITERAL   SWAP [COMPILE] LITERAL [COMPILE] LITERAL ;
    \IF    IMMEDIATE

NEED S>D  \IF   : S>D   ( n - d)     DUP 0< ;
NEED WITHIN    \IF   : WITHIN ( n n2 n3 - f)  OVER - >R - R> U< ;
NEED TRUE \IF   -1 CONSTANT TRUE
\ String operators     See  DDJ  December 1987
\ Only  /STRING  and  EVAL  are used in this application.

: /STRING ( a n n2 - a+n2 n-n2)    ROT OVER +  ROT ROT - ;
\ truncates leftmost n chars of string.  n may be negative.

: EVAL ( a n )
```

```
\ evaluates ("text interprets") a string.
   DUP >R  TIB SWAP CMOVE  R@ #TIB !
   0 >IN ! 0 BLK !  INTERPRET  R> >IN ! ;



\\ String operators from  STRINGS.ARC  are summarized here:

ASCII ( - c)        \ returns value of following character.
CTO"" ( c - a 1)    \ converts character to string.

SKIP ( a 1 c - a2 12)
\ returns shorter string from first position unequal to byte.
SCAN ( a 1 byte - a2 12)
\ returns shorter string from first position equal to byte.

" ( - a n)      \ STATE-smart string literal.



\\ String operators from  STRINGS.ARC  continue here:
VAL? ( a n - d 2 , n2 1 , 0)
\ string to number conversion primitive.  True if d is valid.
\ Returns d if number contains ",-./:"  and sets DPL = 0
\ Returns n if no punctuation present   and sets DPL = 0<

VAL ( a n - d f)
\ converts string to double number.  True if number is valid.
\ If number contains ",-./:" then sets DPL = 0
\ If no punctuation present  then sets DPL = 0<

-TEXT ( a n a2 - -1 , 0 , 1)
\ returns -1 if string a n < a2 n , 0 if equal, and 1 if >.

COMPARE  ( a n a2 n2 - -1 , 0 , 1)
\ returns -1 if a n < a2 n2 , 0 if equal, and 1 if >.
\ The corrected version of  MATCH

: MATCH ( a n a2 n2 - ???? 0 , offset -1)
\ returns the position of string a2 n2 in (a n).
\ Offset is zero if ( a n ) is found in first char position.
\ Returns false with invalid offset if ( a n ) isn't in a2 n2.
   DUP 0= IF  2DROP 2DROP   0 TRUE EXIT  THEN
   2SWAP  2 PICK OVER SWAP -
   DUP 0< IF  2DROP 2DROP   0 EXIT  THEN
   0   ( index ) SWAP 1+ 0
   DO  ( index ) >R
     2OVER 2OVER DROP  -TEXT 0=  ( equal? )
     IF  2DROP 2DROP  R> TRUE  UNDO EXIT  THEN
     1 /STRING   R> 1+
   LOOP  2DROP 2DROP  0 ;

\ Data stream general support
1024 CONSTANT 1K

: UMIN ( u u2 - u3)   2DUP U< 0= IF  SWAP  THEN  DROP ;

\ Adjust these constants for your system:
10 CONSTANT #LF     \ linefeed character.
13 CONSTANT #EOL    \ end-of-line character.
26 CONSTANT #EOF    \ end of file character (control-Z).

\ Adjust end-of-line and end-of-file sequence for your system:
CREATE ENDLINE   2 ( count) C,  #EOL C, #LF C,
CREATE ENDFILE   1 ( count) C,  #EOF C,



\ File size and position
\ Example of some of the structure of a file control block:

\ VARIABLE FCB   HERE FCB !  5 CELLS ALLOT  ( Containing: )
\     1 cell   current file handle-- ie selects current file.
\     2 cells  current file size in bytes (double number).
\     2 cells  current file position      (double number).

\\ You can implement CAPACITY and POSITION as 2VARIABLES.
\\ You must initialize CAPACITY to the size of your file.

2VARIABLE POSITION
2VARIABLE CAPACITY  ( eg  DSIZE CAPACITY 2! )



\ Set and reset file position
\ Given POSITION you can control the position of file access:

: MARKDATA ( - d)   POSITION 2@ ;
\ determines the position of the current file.

: SEEKDATA ( d )    POSITION 2! ;
\ changes the position of the current file.









\ Extend the file
```

*(continued on next page)*

# THE FORTH COLUMN

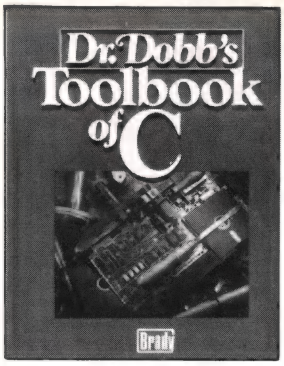
## Listing One

*(Listing continued, text begins on page 108.)*

```
\ If your Forth or operating system requires explicit extension,
\ supply an appropriate definition for  EXTEND .
\ Otherwise, use  : EXTEND ( d ) COMPILE 2DROP ;  IMMEDIATE

: EXTEND ( d ) COMPILE 2DROP ;  IMMEDIATE

\\
: EXTEND ( d )
\ properly extends current file by  d  bytes.
\ This example converts  d  to blocks and calls a MORE function.
   1K UM/MOD  SWAP IF  1+  THEN ( # of blocks to extend ) MORE ;



\ Read and write data files directly

: GETDATA ( a n - n2)   ;
\ reads n bytes of data from input file into address, n < 64K
\ Returns n2 bytes not read ( ie beyond end of file ).
\ Implement as a system call using CAPACITY and POSITION

: PUTDATA ( a n)   ;
\ writes n bytes of data to output file from address, n < 64K
\ Implement as a system call using CAPACITY POSITION and EXTEND



\ Read BLOCKed file as data file

: GETDATA ( a n - n2)
\ reads n bytes of data from input file into address, n < 64K
\ Returns n2 bytes not read ( ie beyond end of file ).
   ( calculate # of bytes to move < 64K : )  POSITION 2@
   BEGIN  2 PICK ( n ) DUP
     IF ( n ) >R  2DUP 1K UM/MOD  SWAP DROP  1+  1K UM*
        CAPACITY 2@ DMIN  2OVER D- 0= NOT OR  R> UMIN
     THEN  ?DUP
   WHILE  >R  2DUP 1K UM/MOD  BLOCK +  4 PICK R@ CMOVE
     R@ 0 D+  2SWAP  R> /STRING  2SWAP
   REPEAT  POSITION 2!  SWAP DROP ;



\ Write BLOCKed file as data file

: PUTDATA ( a n)
\ writes n bytes of data to output file from address, n < 64K
   ( extend the file as needed : )
   DUP 0  POSITION 2@ D+  CAPACITY 2@  2SWAP D-  DUP 0<
   IF  2DUP EXTEND  2OVER CAPACITY 2!  THEN  2DROP
   ( calculate # of bytes to move < 64K : )  POSITION 2@
   BEGIN  2 PICK ( n ) DUP
     IF ( n ) >R  2DUP 1K UM/MOD  SWAP DROP  1+  1K UM*
        CAPACITY 2@ DMIN  2OVER D- 0= NOT OR  R> UMIN
     THEN  ?DUP
   WHILE  >R  2DUP 1K UM/MOD  BLOCK +  4 PICK SWAP R@ CMOVE
     R@ 0 D+  2SWAP  R> /STRING  2SWAP   UPDATE
   REPEAT  POSITION 2!  2DROP ;

\ Read text file with #EOF

: GETTEXT ( a n - n2 f)
\ reads n bytes of text from input file into address, n < 64K
\ Returns n2 bytes not read ( ie end-of-line or beyond file)
\ Returns true if #EOL terminates line; false otherwise.
   POSITION 2@  CAPACITY 2@  2OVER D- 0= NOT OR ( limit to 64K)
   3 PICK UMIN  ?DUP 0= IF  2DROP SWAP DROP  0 EXIT  THEN  0
   DO  2DUP 1 0 D+  2SWAP 1K UM/MOD  BLOCK + C@ ( read a char )
      DUP #EOL = OVER #EOF = OR
      IF  >R  POSITION 2!  SWAP DROP  R> #EOL = UNDO EXIT  THEN
      DUP #LF -  ( a n dpos ch f )
      IF  >R  2SWAP  R@ 2 PICK C!  1 /STRING  2SWAP  R> THEN
      DROP
   LOOP  POSITION 2!  SWAP DROP  0 ;

\ Read text file without #EOF

: GETTEXT ( a n - n2 f)
\ reads n bytes of text from input file into address, n < 64K
\ Returns n2 bytes not read ( ie end-of-line or beyond file)
\ Returns true if #EOL terminates line; false otherwise.
   POSITION 2@  CAPACITY 2@  2OVER D- 0= NOT OR ( limit to 64K)
   3 PICK UMIN  ?DUP 0= IF  2DROP SWAP DROP  0 EXIT  THEN  0
   DO  2DUP 1 0 D+  2SWAP 1K UM/MOD  BLOCK + C@ ( read a char )
      DUP #EOL =
      IF  >R  POSITION 2!  SWAP DROP  R> #EOL = UNDO EXIT  THEN
      DUP #LF -  ( a n dpos ch f )
      IF  >R  2SWAP  R@ 2 PICK C!  1 /STRING  2SWAP  R> THEN
      DROP
   LOOP  POSITION 2!  SWAP DROP  0 ;

\ Read and write lines of text

: GETLINE ( a n - a n2 f)
\ reads n bytes of text from input file into address, n < 64K
\ n2 bytes are actually read; this is the opposite of GETTEXT
\ Returns true if #EOL terminates line; false otherwise.
   2DUP GETTEXT >R  -  DUP 0= 0=  R> OR ;

: PUTLINE ( a n )   PUTDATA  ENDLINE COUNT PUTDATA ;
\ writes n bytes of data to output file from address, n < 64K
```

```
\ Text stream examples

: TYPE-FILE   \ reads and prints the input text file.
\ Assumes zero-length string TYPEs nothing.
   SWITCH ( to input file saving currently active file)
   BEGIN  PAD 80  GETLINE ( n2 f)
   WHILE  CR TYPE  REPEAT  2DROP
   SWITCH ( back to current file) ;

: COPY-FILE
\ copies the input text file to the output text file.
\ Save and restore current file as needed.
   BEGIN  SWITCH ( to input  file)  PAD 80 GETLINE
          SWITCH ( to output file)
   WHILE  PUTLINE  REPEAT  2DROP  ENDFILE COUNT PUTDATA ;

\ Text stream examples
: BLOCK-TO-TEXT
\ copies the input BLOCK file to the output text file.
\ Save and restore current file as needed.
   BEGIN  SWITCH ( to input  file)  PAD 64 GETLINE
          SWITCH ( to output file)
   WHILE  -TRAILING  PUTLINE
   REPEAT  2DROP  ENDFILE COUNT PUTDATA ;

: TEXT-TO-BLOCK   0 ( previous line length )
\ copies the input text file to the output BLOCK file.
   BEGIN  SWITCH ( to input  file)
          PAD  64  2DUP BLANK  GETLINE  ROT ( a ) DROP
          SWITCH ( to output file)
   WHILE  DUP 0= ROT 64 = AND NOT IF  PAD 64 PUTDATA  THEN
   REPEAT  2DROP ;
\ Text stream examples

: EVAL-FILE   \ reads and interprets the input text file.
\ Assumes zero-length interpreted string does nothing.
   SWITCH ( to input file saving currently active file)
   BEGIN  PAD 80  GETLINE ( n2 f)
   WHILE  EVAL  REPEAT  2DROP
   SWITCH ( back to current file) ;
```

**End Listing**




## Dr. Dobb's Toolbook of C

by the Editors of *Dr. Dobb's Journal of Software Tools*

This authoritative reference contains over 700 pages of the best C articles and source code from *Dr. Dobb's Journal of Software Tools*, along with new material by C experts. You'll find hundreds of pages of useful C source code, including a complete compiler, an assembler, and text-processing utilities.

**Book** **Item #615-3** **$29.95**

# Special Packages

## CP/M C Package SAVE $27!

Receive: Dr. Dobb's Toolbook of C, The Small-C Handbook, the Small-C Compiler, Small-Mac Assembler, and Small-Tools Text Processing Programs. Only $99.95!

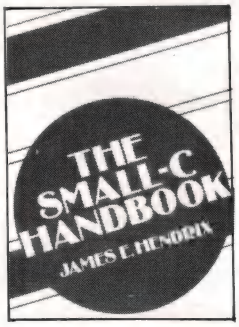
**CP/M C Package** **Item #005A** **$99.95**

## MS-DOS C Package SAVE $22!

Receive: Dr. Dobb's Toolbook of C, The Small-C Handbook and MS-DOS Addendum, the Small-C Compiler, Small-Windows, and Small-Tools Text Processing Programs. Only $109.95!

**MS-DOS C Package** **Item #005W** **$109.95**

**C Disk Formats: Please specify MS-DOS or CP/M. For CP/M, specify: Apple, Kaypro, Osborne, Zenith Z-100 DS/DD, 8" SS/SD**

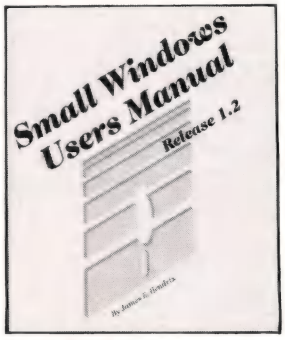



## The Small-C Compiler and Small-C Handbook

by James E.Hendrix

This compiler and handbook provide everything you need for learning how compilers are constructed, and for learning C at its most fundamental level. You'll find a discussion of assembly language concepts and program translation tools, and learn how to generate a new version of the compiler. Full source code is included. The handbook and compiler on disk are available for both MS-DOS and CP/M systems. The handbook comes with an addendum for the MS-DOS version. Please specify format.

**Book & Disk (MS-DOS)** **Item #76-3** **$42.90**
**Book & Disk (CP/M)** **Item #67-4** **$37.90**

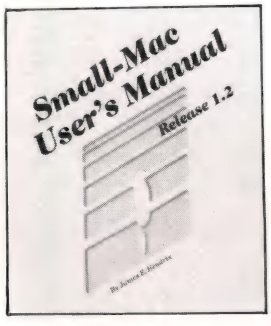



## Small-Windows: A Library of Windowing Functions for the C Language

by James E. Hendrix

**Small-Windows** is a complete windowing library for C (Microsoft 4.0 and Small-C). The package includes: 18 video functions written in assembly language, 7 menu functions that support both static and pop-up menus, 41 window functions to clean, frame, move, hide, show, scroll, push, and pop windows. Two test programs are provided as examples to show you how to use the library and the window, menu, and directory functions. Documentation and full C source code is included. Available for MS-DOS systems for the following compilers: Microsoft C Version 4.0, Small-C, Lattice C and Turbo C. Please specify compiler format.

**Manual & Disk (MS-DOS)** **Item #35-6** **$29.95**

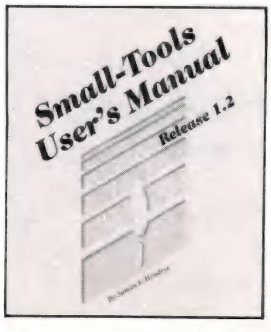



## Small-Mac: An Assembler for Small-C

by James E. Hendrix

This assembler features simplicity, portability, adaptability, and educational value. The package includes: a simplified macro facility; C language expression operators; object file visibility; descriptive error messages; an externally defined instruction table. You get the macro assembler, linkage editor, load-and-go loader, library manager, CPU configuration utility, and a utility to dump relocatable files. Documentation is included. For CP/M systems only. Please specify format.

**Manual & Disk (CP/M)** **Item #77-1** **$29.95**



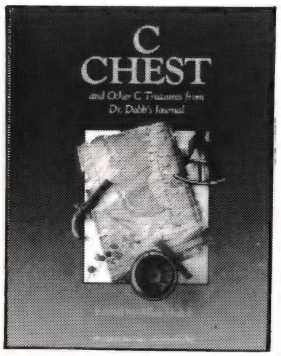

## Small-Tools: Programs for Text Processing

by James E. Hendrix

This package of programs performs specific, modular operations on text files, including: editing; formatting; sorting; merging; listing; printing; searching; changing; transliterating; copying; concatenating; encrypting and decrypting; replacing spaces with tabs and tabs with spaces; counting characters, words, or lines; and selecting printer fonts. Small-Tools is supplied in source code form. With the Small-C Compiler, you can select and adapt these tools to your own purposes. Documentation is also included. Available for MS-DOS or CP/M systems. Please specify format.

**Manual & Disk (MS-DOS or CP/M)** **Item #78-X** **$29.95**

# C Tools From M&T Books




## C Chest and Other C Treasures from Dr. Dobb's Journal

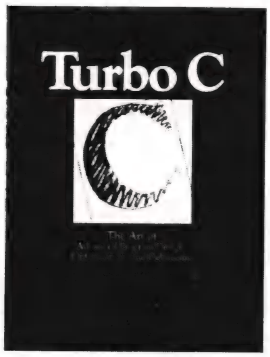
by Allen Holub

This comprehensive anthology contains the popular "C Chest" columns from *Dr. Dobb's Journal of Software Tools*, along with the lively philosophical and practical discussions they inspired, in addition to other information-packed articles by C experts.

Topics covered include: pipes, wild-card expansion, and quoted arguments; sorting routines; command-line processing; queues and bit maps; utilities such as *ls*, *make*, and *more*; expression parsing; hyphenation; IBM cursor control and an Fget that edits; redirection; accessing IBM video display memory; trees; an AVL tree database package; directory traversal; sets; shrinking .EXE file images; hashing, expressions, and Roman numerals; and statistical applications of digital low-pass filters.

All subroutines and programs are written in C and are available on disk with full C source code. MS-DOS format.

| | | |
|---|---|---|
| **Book & Disk (MS-DOS)** | **Item #49-6** | **$39.95** |
| **Book** | **Item #40-2** | **$24.95** |




## Turbo C: The Art Program Design, of Advanced Optimization and Debugging

by Stephen R. Davis

Overflowing with example programs this book fully describes the techniques necessary to skillfully program, optimize and debug in Turbo C. Every topic and Turbo C feature discussed is fully demonstrated in Turbo C source code examples. Advanced topics such as pointers; direct screen I/O; inline statements in Turbo C; and how to intercept and redirect BIOS calls are all covered indepth. The author further demostrates these advanced topics by writing a RAM resident pop-up program in Turbo C. Learn about the differences between Unix C and Turbo C, about the transition from Turbo Pascal to Turbo C, and about the superset of K&R C features implemented in Turbo C and included in the proposed ANSI C standard.

| | | |
|---|---|---|
| **Book & Disk (MS-DOS)** | **Item #45-3** | **$39.95** |
| **Book** | **Item #38-0** | **$24.95** |

## Special Packages Save 15%

Receive the **C Chest** book & disk, the **Turbo C** book & disk, and the **Small-Windows** manual & disk, all for only $93.95! You save 15%

**C Chest/Turbo/Window Package** **Item #168** **$93.95**

## To Order:

Return this order form with your payment to: M&T Books, 501 Galveston Dr., Redwood City, CA 94063

**Or, CALL TOLL-FREE 800-533-4372**

Mon-Fri 8AM-5PM

(In CA call 800-356-2002)

## ORDER FORM

Name _____

Address _____

City _____ State ____ Zip _____

| Item # | Description | Price |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Subtotal _____

CA residents add sales tax ____ % _____

Add $2.25 per item for shipping _____

TOTAL _____

**For Disk Orders,** please indicate format. Refer to product description for standard format availability.

☐ MS-DOS CP/M: ☐ Kaypro ☐ 8" SS/SD ☐ Osborne ☐ Apple ☐ Zenith Z-100 DS/DD

For Small-Windows, indicate:

☐ Microsoft version 4.0 compiler ☐ Lattice C 3.0 compiler

☐ Small-C compiler ☐ Turbo C compiler

☐ Check enclosed. **Make payable to M&T Publishing.**

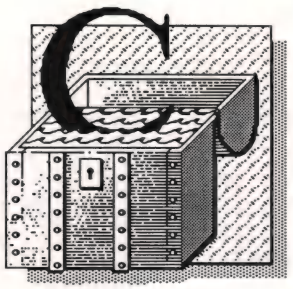
Charge my ☐ VISA ☐ M/C ☐ Am. Ex.

Card No. _____ Exp. _____

3136B

# C CHEST

## Hiding Configuration Information



Storing a program's configuration information—variables that can be changed by users but that must retain their values between successive program invocations—is a common problem. An easy solution is to use a configuration file that's read in by the program when it boots.

Configuration files have problems, though. The first one is finding the file. Many programs require the configuration file to be in the current directory. If you're going to execute the program in more than one directory, you need a configuration file in each of these directories. Clearly this behavior isn't really acceptable. Not only do you start filling up your disk with unnecessary files but also the files can get out of sync with each other—if you make a change to one, you have to make a change to all of them.

There are other solutions, however. First, you can search for the file along the *PATH*. Example 1, page 95, shows a search-for-file subroutine that can be used for this purpose. It's passed two strings—the first containing a file name, and the second the name of an environment that holds the search path (a semicolon-delimited list of directories). The *search( )* subroutine looks for the file, first in the current directory, and then in all the directories listed in the given environment string. If the file is found, *search( )* returns a pointer to the full path name; otherwise it returns *NULL*.

by Allen Holub

The *access( )* subroutine that *search( )* uses is a Unix function that looks at the permission mask associated with a file. You can use it to test for read permission, write permission, and so forth. Here, I'm just using it to test for existence. If you don't have an *access( )* function, you can do the same thing by trying to *open( )* the file for read and looking to see if *open( )* returned an error (if it did, the file didn't exist).

The *strpbrk (char *src, char *pat)* function searches the *src* string for any of the characters in the *pat* string and returns a pointer to that character if found (*NULL* if not). The *strtok (char *src, char *delim)* function extracts a series of tokens from the *src* string. Tokens are delimited by any of the characters in the *delim* string. The first time the subroutine is called, it returns the first token from the string. In subsequent calls, the first argument is set to NULL, and it returns subsequent tokens from the original string. It returns NULL when there are no more tokens.

The *getenv( )* function returns the contents of the indicated environment. This string has to be copied to *pbuf[ ]* because it's modified by the subsequent *strtok( )* calls. *Strpbrk( )*, *strtok( )*, and *getenv( )* are all ANSI functions, so they should be in your compiler's library.

Another alternative to searching along the *PATH* is provided by some compilers, such as Microsoft's. These compilers provide the full path name of the executable file in *argv[0]*. You can then require the configuration file to be in the directory as the executable file.

A third (and, I think, the best) alternative is to dispense with the configuration file entirely and to incorporate the configuration information in the .EXE file itself. This way, you don't clutter up the disk with needless files whose immediate purpose is not obvious. To use the .EXE file, you have to find it on the disk, using either of the methods discussed earlier. You also have to declare a structure in your program that will contain the modifiable options. At very least this structure must contain a signature field and a checksum. The signature contains an arbitrary, but unchanging, string that you can look at to see if the options are valid.

A stripped-down options structure (called *Opts*) is shown in Example 2, page 96. The Microsoft compiler correctly evaluates *sizeof(DEF_SIG)* as the number of characters in the string (including the \0). I can't vouch for other compilers, though. Because the length is used in a declaration, you can use a *strlen( )* call to compute it (because it's executed at run time, not compile time). Consequently, if your compiler's *sizeof* doesn't work correctly, you'll have to count the characters in the signature string to declare the array.

Options are fetched from the .EXE file with a call to *get_opts( argv[0] )* at the head of my *main( )* subroutine. Here, *argv[0]* holds the full path name of the .EXE file. If this is the first time that the program is executed, *get_opts( )* creates (and initializes) the options area in the file. *Get_opts( )* is shown in Example 3, page 96.

Options are stored at the end of the .EXE file, following any executable code. The .EXE file is opened for binary-mode read on line 14. I then seek to what ought to be the beginning of the options area on line 20. That is, the options are at the end of the file, so I seek to end of file less the size of the *Opt* structure. The structure is loaded on line 22. Now I look at the signature (on line 25). If it doesn't match, I assume that this is the first time that the program has been run, in which

# 10 Important Reasons C Programmers Use Our File Manager

### 1. It's written in C.
Clearly the growing language of choice for applications that are fast, portable and efficient. All of db_VISTA's source code is written in C.

### 2. It's fast – almost 3 times faster than a leading competitor.
Fast access that comes from the unique combination of the B-tree indexing method and the "network" or direct "set" relationships between records. A winning combination for fast performance.

### 3. It's flexible.
Because of db_VISTA's combination of access methods, you can program to your application needs with ultimate design flexibility. Use db_VISTA as an ISAM file manager or to design database applications. You decide how to optimize run-time performance. No other tool gives you this flexibility without sacrificing performance.
db_VISTA is also well behaved to work with most any other C libraries!

### 4. It's portable.
db_VISTA operates on most popular computers and operating systems like UNIX, MS-DOS and VMS. You can write applications for micros, minis, or even mainframes.

### 5. Complete Source Code available.
We make our entire C Source Code available so you can optimize performance or port to new environments yourself.

### 6. It uses space efficiently.
db_VISTA lets you precisely define relationships to minimize redundant data. It is non-RAM resident; only those functions necessary for operation become part of the run-time program.

### 7. Royalty free run-time.
Whether you're developing applications for yourself or for thousands, you pay for db_VISTA or db_QUERY only once. If you currently pay royalties to someone else for your hard work, isn't it time you switched to royalty-free db_VISTA?

**db_VISTA™**

**Features**
- **Multi-user** support allows flexibility to run on local area networks
- **File structure** is based on the B-tree indexing method
- **Transaction processing** assures multi-user consistency
- **File locking** support provides read and write locks
- **SQL-based db_QUERY** is linkable
- **File transfer** utilities included for ASCII, dBASE optional
- **Royalty-free** run-time distribution
- **Source Code** available
- **Data Definition Language** for specifying the content and organization of your files
- **Interactive database access** utility
- **Database consistency check** utility

**File Management Record and File Sizes**
- Maximum record length limited only by accessible RAM
- Maximum records per file is 16,777,215
- Maximum file size limited only by available disk storage
- Maximum of 256 index and data files
- Key length maximum 246 bytes
- No limit on number of key fields per record
- No limit on maximum number of fields per record

**Operating System & Compiler Support**
- **Operating systems:** MS-DOS, UNIX, XENIX, ULTRIX, Microport, VMS, Macintosh
- **C compilers:** Lattice, Microsoft, IBM, Aztec, Turbo C, XENIX, UNIX and LightspeedC

### 8. db_QUERY & db_REVISE.
Add the SQL-based, ad hoc query and report writer for a relational view of db_VISTA databases.
Use db_REVISE to re-design your database easily and quickly!
Both royalty free!

### 9. Free tech support.
60 days of free technical and application development support for every Raima product. Of course, extended support and training classes are also available at your place or ours.

### 10. Upward database compatibility
Start out with file management in a single-user PC environment—then move up to a multi-user LAN or a VAX database application with millions of records. You'll still be using db_VISTA. That's why so many C programmers are choosing db_VISTA.

John, You forgot one...
11. db_Vista training class
All you need to know to get the most from db_Vista! 2-5 days...Basics, Advanced, and Internals.
Call Now! Will cut development costs a lot!
April 11-15

### 30-day Money Back Guarantee!
Try db_VISTA in your environment for 30 days and prove it to yourself. If not completely satisfied, return it for a

### Price Schedule

| | db_VISTA | db_QUERY |
|---|---|---|
| ☐ Single user | $ 195 | $ 195 |
| ☐ Single user w/Source | $ 495 | $ 495 |
| ☐ Multi-user | $ 495 | $ 495 |
| ☐ Multi-user w/Source | $ 990 | $ 990 |
| **NEW:** | | |
| ☐ VAX Multi-user | $ 990 | $ 990 |
| ☐ VAX Multi-user w/Source | $1980 | $1980 |

### Order Now.
Put db_VISTA to work in your application program. Ordering is easy—simply call toll-free. We'll answer your technical questions and get you started. Call today.

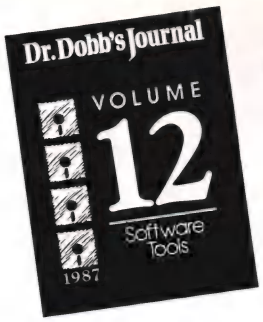
## Call Toll-Free Today!
**1 (800) db-RAIMA**
(800/327-2462) or
206/828-4636

MasterCard VISA

RAIMA™ CORPORATION

3055 - 112th NE, Bellevue, WA 98004 USA
(206) 828-4636 Telex: 6503018237 MCI UW

DDJ 2/88




# Dr. Dobb's Bound Volume 12

~~$39.95~~ **$35.75**

## FULL 12 VOLUME SET

~~$403.~~ **$345.**

**FULL 12 VOLUME SET** **$345.00**

Receive the entire 12 volume set of **Dr. Dobb's Bound Volumes**, including 12 years worth of useful source code for only $345! You save over $60!

**Item # 93-3** ~~$403.~~ **$345**

**Volume 12 - 1987**

Building better brains. DDJ broke new ground in 1987 with a neural network implementation, monthly coverage of artificial intelligence and object-oriented programming techniques, and reviews of implementations of Prolog, LISP, and Smalltalk. We also evaluated the new BASICs, the proposed ANSI C standard, and the new optimizing C compilers, and we delivered practical utilities in C, Pascal, Forth, and assembly language. We gave our readers a UNIX BBS, an implementation of the nroff text editor, an extension to Appletalk, and an extended COM port driver. DDJ also described how 2-dimensional graphics can be displayed in 3-D. We showed how to create Macintosh Buttons, Amiga Gadgets, and DOS device drivers. And, as always, we delivered more useful code than any other magazine.

**Item # 84-4** ~~$39.95~~ **$35.75**

**Volume 1 - 1976**

Always pertinent for bit crunching and byte saving, home-brew computer projects, and the technical history of home computing. Topics include: Tiny BASIC, the first words on CP/M, speech synthesis, floating point routines, the 6502 disassembler for the Apple, and much more.

**Item #13-5** **$30.75**

**Volume 2 - 1977**

The small computer emerges as a powerful tool for the modern age in these issues from 1977. Topics include: Lawrence Livermore Lab's BASIC, Dr. Starkweather's PILOT, using a modem, string- handling techniques, ciphers, Turtle graphics, and micro utilities.

**Item #16-X** **$30.75**

**Volume 3 - 1978**

Explores the foundation of the mass-market computer industry in 1978. Topics include: programming in BASIC, PILOT, and Pascal; RAM memory testers; Apple utility programs; the S-100 bus standard; STRUBAL; and a structured BASIC compiler.

**Item #17-8** **$30.75**

**Volume 4 - 1979**

Innovative ideas and articles guide the reader through the age of discovery in 1979. Topics include: selecting business software, microcomputer speech and music, information networks, and interfacing techniques.

**Item #14-3** **$30.75**

**Volume 5 - 1980**

Focuses on the technological promise of the modern microcomputer. Topics include: the revolutionary impact of CP/M, C programming and the UNIX operating systems, a survey of computer networks, software portability, introduction to Forth, and compiler writing.

**Item #18-6** **$30.75**

**Volume 6 - 1981**

The microcomputer enters the mainstream. Topics include: computer conferencing, the power of Forth, 8- and 16-bit technology, Rubik's cube simulator, and an adventure game development system.

**Item #19-4** **$30.75**

**Volume 7 - 1982**

Examines the potential of powerful 16-bit micros. Topics include: in-depth coverage of Forth, 68000, 8088 programming, C software tools, utilities for CP/M - including a spelling checker, using bulletin boards, and more.

**Item #20-8** **$35.75**

**Volume 8 - 1983**

DDJ turns pro. Some of the most powerful professional programmer's tools ever published in a magazine are in this 1983 volume, including Small-C, the RED editor, and an Ada subset.

**Item #00-3** **$35.75**

**Volume 10 - 1985**

The year of living dangerously. In 1985 DDJ added more memory, an SCSI port, and a hard disk to the Macintosh; challenged the UNIX establishment with plans for a free UNIX. Includes software tools in C, Modula-2, Forth, Pascal, assembly language, and Prolog.

**Item #21-6** **$35.75**

**Volume 9 - 1984**

In 1984 DDJ examined new programming environments Prolog, expert systems, Modula-2, and Pascal. Other topics include GREP, UNIX internals, and two encryptions systems.

**Item #08-9** **$35.75**

**Volume 11 - 1986**

The promise of power. DDJ covered 1986's changes with issues on the 68000, parallel processing, artificial intelligence, the 80386, and multitasking. DDJ also supported the new chips with assemblers, translators, and other development tools.

**Item #31-3** **$35.75**

## To Order:

Return this Order Form with your payment to:
M&T Books, 501 Galveston Drive,
Redwood City, CA 94063.
Or, **CALL TOLL-FREE 800-533-4372**
(in CA 800-356-2002.)

Name _____

Address _____

City _____ State _____ Zip _____

| | | | |
|---|---|---|---|
| ☐ Vol. 1 | $30.75 _____ | ☐ Vol. 7 | $35.75 _____ |
| ☐ Vol. 2 | $30.75 _____ | ☐ Vol. 8 | $35.75 _____ |
| ☐ Vol. 3 | $30.75 _____ | ☐ Vol. 9 | $35.75 _____ |
| ☐ Vol. 4 | $30.75 _____ | ☐ Vol. 10 | $35.75 _____ |
| ☐ Vol. 5 | $30.75 _____ | ☐ Vol. 11 | $35.75 _____ |
| ☐ Vol. 6 | $30.75 _____ | ☐ Vol. 12 | $35.75 _____ |
| (In U.S., add $3.25 per book, $22 per 12 Volume set) | | ☐ Vol. 1-12 | $345 _____ |
| (Outside U.S., add $6.75 per book foreign surface) | | | Subtotal _____ |
| | | CA residents add approporiate sales tax | _____ % _____ |
| | | | Shipping _____ |
| ☐ Check enclosed. **Make payable to M&T Books.** | | | TOTAL _____ |

**Charge my** ☐ VISA ☐ M/C ☐ Amer. Exp.

Card # _____ Exp.Date _____

**3136A**

*(continued from page 92)*

case the options won't exist yet. So, I initialize the *Opt* structure myself on lines 27–38.

A checksum for the structure is computed on line 39–42. The checksum is the negative sum of the other bytes in the structure. That is, if you add up the contents of the entire area occupied by the structure (including the checksum), you'll get 0. The checksum has two purposes: obviously, it can be used for checking the validity of the data in the structure itself; it also keeps DOS happy because every .EXE file also has a checksum. The structure's local checksum cancels any effect that the rest of the structure's contents would have on the .EXE file checksum. That is, because the sum of all the bytes in the structure is 0, the presence or modification of the structure won't change the file's checksum.

I seek to end of file and write out the initialized options area on lines 43–48. I *#ifndef*ed out the actual *write( )* call when I'm debugging because CodeView, as it also puts stuff at the end of the .EXE file, gets confused if it can't find its own information there. That is, it thinks that there's no debugging information if I add my own data to the end of the file.

If any options were changed during the run, the options area in the .EXE file is updated when the program terminates with a call to *put_opts(argv[0])*, where again *argv[0]* holds the full path name of the .EXE file. *Put_opts( )* is shown in Example 4, page 96. It opens the .EXE file and then recomputes the checksum to reflect the changes made during the run. The routine then seeks in the file to the beginning of the options space (which must exist) and writes out the modified structure. Again, I *#ifdef* out the actual write if I'm debugging to keep CodeView happy.

### *Nifty Stuff: awk*

awk, in addition to being a flightless, seagoing bird and the sound made by that bird, is one of the more useful tools in the Unix toolbox. Many (perhaps most) programs do

```
#include <stdio.h>

extern char *strpbrk(),          /* Library routines */
            *strtok(),
            *getenv();

static  char    *search( file, env_name )
char    *file, *env_name ;
{
    /* Search for file by looking in the directories
     * listed in the env_name environment.
     * Return a pointer to the full path name if you
     * find it (NULL if you don't). The returned string
     * is transient; it will be modified by the next

     * call to search().
     */

    static   char       pathname[80];
    char                pbuf[129];
    char                *p ;

    strcpy( pathname, file );

    if( access( pathname, 0 ) != -1 )
        return pathname;

    /* The file doesn't exist in the current directory.
     * If a specific path was requested (ie. if
     * file contains the characters \ or /) or if
     * the PATH environment isn't set, return a NULL;
     * else search for the file along the path.
     */

    if( strpbrk(file,"\\/")  ||  !(p = getenv(env_name)) )
        return NULL;

    strncpy( pbuf, p, 129 );

    if( p = strtok( pbuf, ";" ) )
    {
        do
        {
            sprintf(pathname, "%0.50s\\%0.20s",
                              p, file );

            if( access( pathname, 0 ) >= 0 )
                return pathname;
        }
        while( p = strtok( NULL, ";") );
    }
     return NULL;
}
```

***Example 1:*** *Search( )*

*(continued from page 95)*

nothing but manipulate or verify data. Filter programs, such as grep, sed, pr, and so forth, just output a shuffled around version of an input file. Other programs, such as database report generators, shuffle around a database and output parts of it in an ASCII representation. Though none of these programs are particularly hard to write, it's a nuisance to write a hoard of special-purpose programs in a language such as C—especially if you're going to use that program once and throw it away.

awk provides a solution to this problem. It is essentially a dialect of C that's optimized for text processing—a general-purpose tool from which other tools can be built. It supports all the C operators and control-flow statements (even recursion), though the operators have been extended to work with strings. For example:

if( "aardvark" < "zebra" )

evaluates to true.

awk programs all take the form *expression* { *action* }, where the expression tells awk when to apply the action. That is, the action can be applied on every line, on a range of lines, a group of lines delimited by a specific string, on every line that contains a match for a regular expression, on every line that has a specific string or number in a specific field, and so forth. The action can be a simple *printf( )* statement (*printf( )* is an awk primitive) or a complex program that does elaborate database manipulation. Most of the familiar C constructs are available.

As an example, the following simple awk program numbers all

```
#include <stdio.h>
#include <fcntl.h>

#define DEF_SIG  "(C) 1987, Allen I. Holub. All rights reserved."

struct options
{
    char    signature[ sizeof(DEF_SIG) ];
    int     chksum;
    /*************************************************/
    /* This structure also contains fields for every */
    /* option that the user can change at run time.  */
    /*************************************************/
}
Opt ;
```

***Example 2:*** *Options header*

```
get_opts( name )
char    *name;
{
    /* Various statistics (such as last time the program was
     * run) are stored in a buffer at the end of the
     * executable file. These stats are all stored in a
     * "options" structure. This subroutine creates the data
     *  area if it doesn't exist and initializes Opt as
     *  appropriate. /

    int  fd;
    int  i, *p;          /* Used to create checksum */

    if( (fd = open( name , O_RDWR | O_BINARY )) == -1 )
    {
        perror ( name );
        exit   ( 1    );
     }

    lseek ( fd, 0L - sizeof(Opt), SEEK_END      );

    if( read( fd, (char *) &Opt, sizeof(Opt) ) != sizeof(Opt))
        ferr( "Internal error: Can't read options\n" );

    if( strcmp( DEF_SIG, Opt.signature ) != 0 )
    {
        memset( &Opt, 0 , sizeof(Opt)); /* The memset is for
                                         * debugging.(it shows
                                         * us that the record.
                                         * has been written
                                           correctly./

        strcpy ( Opt.signature, DEF_SIG  );

        /*******************************************************/
        /* Initialize other fields in the Opt structure here */
        /*******************************************************/

        Opt.chksum = 0 ;
        for( p = (int *)(&Opt), i = sizeof(Opt)/2; --i >= 0; )
                Opt.chksum -= *p++;

        lseek ( fd, 0L, SEEK_END );

#       ifndef DEBUG
            if( write( fd, (char *) &Opt, sizeof(Opt))
                                          != sizeof(Opt) )
                ferr("Internal error: Can't initialize
                                            options\n");
#       endif
    }

    close( fd );
}
```

***Example 3:*** *Get_opts( )*

```
put_opts( name )
char    *name;
{
    /* Update the options buffer (which better exist).
     * If the options buffer doesn't exist, this
     * subroutine will destroy the end of the file.
     */

    int  fd, *p, i, checksum ;

    if( (fd = open( name , O_RDWR | O_BINARY )) == -1 )
    {
        perror( name );
        exit(1);
    }

    /* Recompute the checksum */

    checksum = 0 ;
    for( p = (int *)(&Opt), i = sizeof(Opt)/2; --i >= 0; )
            checksum -= *p++;


    if( checksum != Opt.chksum )
        printf( "Options have changed" );

    Opt.chksum = checksum;

#   ifndef DEBUG
        lseek ( fd, 0L - sizeof(Opt), SEEK_END );
        if( write(fd, (char *) &Opt, sizeof(Opt))
                                    != sizeof(Opt) )
            ferr( "Can't do final options update\n" );
#   else
        printf("\n***** No I'm not" ****** );
#   endif

    close( fd );
}
```

***Example 4:*** *Put_opts( )*

```
BEGIN { pageno = 0; }

{
    if( (NR % 55) == 0 || pageno == 0 )
    {
        if( pageno )                # not the first page
            printf("\f");

        printf("%s, page %d\n", FILENAME, ++pageno );
        printf("-------------------------------\n\n");
    }
    printf("%3d: %s\n", NR, $0)
}

END { printf("\f"); }
```

***Example 5:*** *Listing.awk, an awk program to create listings*

the lines in the input file and sends the result to standard output:

```
awk '{printf("%3d: %s/', NR, $0)}'
input
```

This one-line program is made more useful in the awk program in listing.awk, Example 5, page 96. When executed with:

```
awk –f listing.awk input
```

a file called input is read and then printed with all the lines numbered. A header giving the file name and page number is printed at the top of every page, and a form feed is printed at the bottom. Looking at the program itself, the *BEGIN* action is done before any input is processed. Here it sets *pageno* to 0. Variables are declared implicitly by using them. The *END* statement is executed at the end of the input. Here it prints a form feed. *NR* and *FILENAME* are predefined variables that hold the current line number and input file name. Because there's no specific pattern or other line selector to the left of the *action*, it's performed on every line. The # delimits a comment.

To go to the other extreme, Example 6, page 98 (extracted from the book discussed later), holds a version of the Unix make utility, written entirely in awk. I've included it here primarily so that you can see the power of the awk programming language. *Getline* is a built-in function that reads a line of input (in this case from *makefile*). Similarly, *sub* substitutes matches of the regular expression given as the left argument with the pattern given as the right argument. *Print*, *printf*, *system*, and *close* are also built-in functions that work as you would expect from their names. *$1*, *$2*, and so on are the fields on the line (fields are space- or tab-delimited by default), and *$0* is the whole line. The ~ *(tilde)* operator is the matches operator, so $0 * /~[A–Za–z]/ checks to see if the first character on the line is a letter.

To my surprise and delight, awk is now available to us masses. *The AWK Programming Language* by Alfred Aho, Brian Kernighan, and Peter Weinberger is an excellent introduction to the language itself, and an executable version called MKS AWK is available for the IBM PC and clones from Mortice Kern Systems (43 Bridgeport Rd. E, Waterloo, Ontario, Canada N2J 2J4) for $75.

The book is both lucid and quite readable (surprising considering the turgid prose in Aho's other books). It starts out with a short tutorial introduction to awk. The next chapter is a complete (but somewhat dry) language description, and the remainder of the book contains awk programs, some quite complex. There are chapters on general data processing; database management and report generation (an awk query language is presented); word-processing applications (for example, an index generator that works with troff); language processing (such as a graph-generating language that takes a graph description as input and outputs an actual graph); and more (for example, a few fancy sort programs, including a heap sort and a topological sort program that works like the Unix tsort, are presented).

I've only one complaint about the book: the actual code is often poorly formatted and undercommented, so some of the examples are difficult to read. *The AWK Programming Language* is much like Kernighan and Ritchie's *The C Programming Language* in this respect. Nonetheless, as in K & R, the examples are often instructive and the time spent deciphering them is well spent.

MKS AWK is a complete implementation of the awk language described in Aho's book. It's quite solid and very much like the Unix System V (Release 3.1) version (some would say too much like Unix—the ubiquitous *syntax error* is printed for virtually every typo that you make in the source file). Four versions of the program are supplied—a large model, with and without an 8087, and a small model, with and without an 8087. In addition, versions of the Unix date, glob, join, sort, and tr programs are provided along with a DOS-specific program that let's you change the switch character used by COMMAND.COM from a / to something more reasonable (thereby letting you use / in path names).

Documentation is supplied in a

5.5- × 8.5-inch saddle-stitched booklet. It is adequate for describing the awk language but is by necessity not as complete as Aho's book. I'd recommend getting both the book and the program, even though the MKS documentation contains everything you need to get started.

In all, awk is a remarkably useful tool. It was the one major Unix utility that I hadn't seen running under DOS, and it's a welcome addition to my toolchest. The Mortice Kern implementation is very good; I recommend it highly.

### Books and File Dumps

Like most people, when it comes to work, I have the best of intentions but rarely manage to get enough accomplished. Consequently, I've an ever-growing stack of books to review gradually taking over what little bare space is available on the top of my desk. This month I'll look at one of these and hopefully clear up the rest of them in upcoming months.

Harbison, Samuel P.; and Steele, Guy L., Jr. *C: A Reference Manual*. 2nd ed. Englewood Cliffs, N.J.: Prentice-Hall, 1987.

Harbison and Steele's book has long been the best reference available on the C language—much better, in fact, than Appendix A of K & R. Prentice-Hall has just published a second edition that makes the book even more valuable than before. The new edition has been updated to include the various ANSI extensions (at least, the extensions as they stood in late 1986). It describes the complete C language in considerable detail—the book was originally intended to be the specification for a compiler project, so it goes into the language with the detail necessary to actually write a compiler—and it also covers all the ANSI library functions. Particularly valuable are discussions of implementation-dependent issues that are likely to affect portability.

There is one unfortunate omission from the second edition. The first edition had two formal grammars for C: one was intended to show you the language syntax; the other was a less readable but more practical grammar, such as you would submit to YACC. The second of these has been left out of the new edition.

Nonetheless, this is a very valuable book that should be in every C programmer's library. I can't recommend it too highly.

### Availability

All the source code for articles in this issue is available on a single disk. To order, send $14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

### Errata:

Last month's column examples contained some errors:

delete all "\sc128\".

Example 3, line 14 should read:

```
int sam( ), dave( ), timer( ), idle( ),
                              maintask( );
```

Example 3, line 35 should read:

```
printf("%ld interrupts, %ld
          blocked\n", t__numint( ),
                              t__numblk( );
```

Example 9, line 7 should read:

```
:(xhat + = (data - xhat) / + +ki)
```

**DDJ**

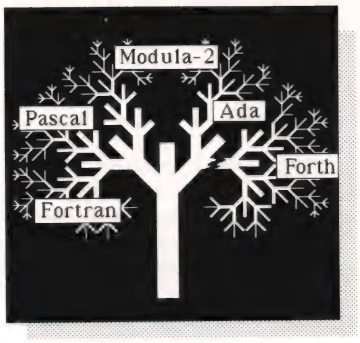
**(Listing begins on page 82.)**



```
#
# A make version of AWK. Taken from The AWK Programming Language
# by Aho, Kernighan, and Weinberger, p. 178.
#
BEGIN {
    while( getline <"makefile" > 0 )            # look for  $1: $2 $3
        if( $0  /^[A-Za-z]/) {
            sub(/:/,"")

            if (++names[nm = $1]] > 1)
                error(nm " is multiply defined")

            for( i = 2; i < NF; i++ )   # remember targets
                slist[nm, ++scnt[nm]] = $i

        } else if ($0  /^\t/)                   # remember cmd for current name
            cmd[nm] = cmd[nm] $0 "\n"

        else if ( NF > 0 )
            error( "illegal line in makefile: " $0 )

    ages()                                      # find initial ages

    if( ARGV[1] in names )
    {
        if (update(argv[1]) == 0 )
            print ARGV[1] " is up to date"
    }
    else
        error( ARGV[1] "is not in makefile" )
}
function ages( f, n, t )
{
    # execute ls -t (which gives a list if files sorted by time last
    # modified) and pipe the result into getline. That is, the for
    # loop process the output from the ls command, one line at a time

    for( t = 1; ("ls -t" | getline f) > 0; t++ )
        age[f] = t                          #all existing files get an age
    close("ls -t")
    for( n in names )
        if( !(n in age) )                   # if n has not been created
            age[]n] = 9999                  # make n really old
}

function update(n, changed, i, s )
{
    if( !(n in age  )) error( n "does not exist" )
    if( !(n in names)) return 0

    changed = 0
    visited[n] = 1

    for( i = 1; i <= scnt[t]; i++ ) {
        if( visited[s = slist[n,i]] == 0)
            update(s)
        else if( visited[s] == 1 )
            error(s "and " n "are circularly defined")

        if( age[s] <= age[n])
            changed++
    }
    visited[n] == 2
    if( changed || scnt[n] == 0 )
    {
        printf("%s", cmd[n])
        system(cmd[n])                      # execute cmd associated with n
        ages()                              # recompute all ages
        age[n] = 0                          # make n very new
        return 1
    }
    return 0
}

function error(s) { print "error: " s ; exit }
```

***Example 6:*** *An awk implementation of make*

# Writing a DOS Critical Error Handler




by Kent Porter

An application running under DOS gets ugly treatment when a hardware error occurs. The alternatives DOS provides are the infamous Ignore (at peril), Retry (and get the same error again), and Abort (losing the work done so far). To get around these nasty consequences, you can write a special kind of interrupt service routine called a critical error handler. That's what I'll do here, and in the process I'll point out some of the features that make the new Turbo Pascal, Version 4.0, a real treat for serious application developers.

The critical error handler wakes up (via interrupt *24h*) when DOS encounters any of 14 problems associated with disk and character devices. These problems run the gamut from an open drive door to media failure. The default critical error handler built into DOS displays a message explaining the problem and asks users to select from the Abort/Retry/Ignore alternatives. Because the default handler sets no state switches and returns nothing, DOS applications that don't replace it can't make any provision for critical errors. The best you can hope for is that the error will go away if the user selects Retry—fat chance.

A better solution is to take over the vector for interrupt *24h*, pointing it to your own handler, which records what went wrong and sets a flag, then returns. Your application can check the flag after each disk or printer I/O request and, if it finds that a critical error occurred, recover or shut down gracefully; the action depends on the nature of the error.

There are two classes of critical errors—disk and nondisk. Although DOS triggers interrupt *24h* for both, they require different processing. On entry, bit 7 of register *AH* contains a 0 for disk errors and a 1 for nondisk problems. Because interpretation of the registers differs from that point on, entry processing should test this bit and branch to the appropriate routine.

Disk errors occur more often than nondisk errors do, and there are more possibilities. Consequently, DOS passes a number of items of information, especially in register *AH*. Bit 0 contains 0 if a read failed and 1 if a write failed. Bits 1 and 2 show where the error was detected, the patterns being:

| | |
|---|---|
| 00 | DOS work area |
| 01 | File allocation table |
| 10 | Disk directory |
| 11 | Files area |

Register *AL* indicates the drive on which the failure took place, the values being 0 for A, 1 for B, and so on. The low half of the *DI* register contains an error code (and the upper half garbage, so isolate the low byte before attempting to interpret the code). Finally, the *BP:SI* register pair points to the device driver header, although nobody cares in handling disk errors. The failure codes returned in *DI* are listed in Table 1, page 103.

The nondisk error class is a catchall for two entirely unrelated kinds of problems. One occurs when a character device—nominally a printer, but it could be a serial port, too—signals a malfunction. The other occurs when DOS detects a discrepancy between the two in-memory copies of the file allocation table (FAT), which governs the management of disk space.

To determine which error is being reported, use the address in *BP:SI* to check the attribute word in the device driver. It's at offset 4 from *BP:SI*. If bit 15 is off, the FAT is corrupted; if it's on, you have a problem with a character device. In the latter case, you can determine which device failed by inspecting the string at offset 8 from *BP:SI*. It gives the logical device name (LPT1, COM1, and so on) of the guilty party.

The critical error handler built into DOS uses this information to produce a brief explanation of the problem. It then asks the user the infamous Abort/Retry/Ignore question and, based on the answer, passes one of three codes back to DOS in the *AL* register. A code of 0 means Ignore (that is, pretend nothing happened and restore control to the running program); 1 means Retry the operation; and 2 means Abort the program without giving it a chance to close files.

Because the purpose of writing a custom error handler is to avoid the catastrophic consequences of these choices, your handler should record information about what went wrong, set a Boolean flag (*CriticalErrorOccurred*) to *TRUE*, and always return 0 in register *AL*. That way DOS will restore control to your program, which can check the flag with the test:

```
if CriticalErrorOccurred then ...
```

and take appropriate action.

And what might that action be? Well, it depends. If the disk is unformatted (error code 8), you might execute *FORMAT* as a child process using Turbo Pascal 4.0's *EXEC* procedure. For drive not ready (code 2), you could tell the user to stick in a disk and close the door, then retry. Unrecoverable errors call for more

# What experts are saying about PC Scheme from Texas Instruments:

*"...Complete, well-designed and inexpensive at $95 ...a complete Lisp implementation for the professional programmer."*

*PC Tech Journal*, August 1986 (Product of the Month)

Discover how powerful—and inexpensive—PC symbolic programming can be with PC Scheme from Texas Instruments. Whether you're an experienced Lisp programmer or just beginning, PC Scheme is the complete, $95* solution to your software development needs.

PC Scheme combines elegant simplicity with remarkable speed in a full Lisp development system. Named *PC Tech Journal*'s Product of the Month (August 1986), PC Scheme brings professional Lisp programming features to personal computers.

**PC Scheme 3.0**

- —Optimizing incremental byte-code compiler for ease of programming and operation
- —EMACS-like editor
- —Lexical scoping of variables
- —Ability to suspend PC Scheme, execute DOS-based programs, then return to PC Scheme
- —Random-file access and binary-file support
- —Extensions for debugging, graphics and windowing
- —External language interface to C, Turbo Pascal® and other languages
- —SCOOPS (Scheme Object-Oriented Programming System)
- —Two-megabyte extended/expanded memory support
- —New manuals with tutorials and examples

Find out for yourself why experts are praising PC Scheme. For the dealer nearest you, or to order by phone, call toll-free:

**1-800-527-3500**

** TI Suggested list price*

*PC Scheme runs on IBM® Personal Computers and compatibles (including the Texas Instruments Business-Pro™ computer). Minimum configuration: 512K RAM, dual floppy system.*

*Turbo Pascal is a registered trademark of Borland International. IBM is a registered trademark of International Business Machines Corporation. Business-Pro is a trademark of Texas Instruments Incorporated.*

TEXAS INSTRUMENTS

261933B

*(continued from page 100)*

drastic action, such as closing all files and terminating the program. The error handler itself only records the bad news; what you do about it is up to you.

There's a more or less ironclad rule among DOS hackers that you shouldn't attempt any I/O from inside an interrupt handler. This is because of the notorious DOS reentrancy problem. An exception is made for a critical error handler, however; it can perform console I/O functions (DOS interrupt *21h*, functions *01h* through *0Ch*). Turbo Pascal and most other high-level languages use these very functions to communicate with the user, so you can safely carry on a console dialog from within the handler.

### *Examining the Code*

Now let's look at the real live critical error handler in Listing One (see page 84). It's written as a unit, a new feature of Turbo Pascal 4.0 that provides for separately compiled, linkable modules.

The only externally visible routine in the *criterr* unit is the *InstallCEH* procedure. It's called by the using program to stuff the address of the critical error handler into the vector for interrupt *24h*, thereby activating the handler. It also initializes the variables set by the handler and determines the location of the video buffer so that the handler can save and restore the display image without permanently corrupting it.

| DI | Means |
|---|---|
| 00h | Write-protected disk |
| 01h | Invalid drive designator |
| 02h | Drive not ready: empty or door open |
| 03h | Unknown command: probably bad DOS call |
| 04h | CRC data error: bad disk |
| 05h | Invalid request structure: program error |
| 06h | Seek error: hardware failure |
| 07h | Unknown media type: probably bad disk |
| 08h | Sector not found: usually unformatted disk |
| 0Ah | Write fault |
| 0Bh | Read fault |
| 0Ch | General failure |

***Table 1:*** *Critical error failure codes returned in* DI

Incidentally, it's not necessary to restore the vector to the default DOS routine when the program ends. That's because DOS automatically does this as part of job termination processing. For that reason, the unit lacks an uninstall procedure.

The handler itself occupies the rest of the unit and is sufficiently generalized to serve as a model. This handler sets four global variables when it gets control: a Boolean flag to indicate that an error occurred, the error code, the drive if a disk, and an action code (the initials of Abort, Retry, and Ignore).

Notice the heading for *CEHandler* in the *Implementation* section of the unit. The *$F+* switch forces far calls, thus guaranteeing that the installation procedure will get a full 32-bit segment:offset pointer to set the interrupt vector. The parameters to *CEHandler* are the general registers. Because of the *Interrupt* keyword following the heading, the compiler saves the named registers on the stack at entry, making their contents available as local variables.

The body of the handler begins

*(continued from page 103)*

after the third local subroutine. It immediately sets the error flag and dissects the *AX* register into its two byte-size components. Next the handler saves the cursor position and copies the screen image onto the heap to prevent it from being corrupted. After determining the error class, it dispatches the appropriate subhandler to report the problem to the user. It then determines what the user wants to do about it (Abort/Retry/Ignore) and records it as the action code. If the user says to Ignore, it resets the four reporting variables. After restoring the screen image and cursor position, it zeros the *AX* register, telling DOS to pay no attention, and returns from the interrupt.

The *DiskError* and *NonDiskError* routines are called from the main body, depending on the error class. Both call the *GiveReason* procedure, which merely prints a diagnostic message on the screen based on the error code passed to it.

*DiskError* is straightforward, loading the error drive global and advising the user of the location and nature of the problem. The *NonDiskError* function is a little more complex because it deals with two distinct kinds of errors. The branch is based on the high-order bit of the device attribute. Note the *Repeat... Until* loop that outputs the device name; it can be up to eight characters long, but if it's shorter, the unused bytes are ASCII 0s. Consequently, the loop halts on a null or after the eighth character, whichever comes first.

Strung throughout the handler are instructions that set the appropriate error indicators so that the using program can sense what went haywire and decide what corrective action to take.

### *Testing It Out*

Make sure the door to drive A is open when you run the cerrtest program in Listing Two, page 85. The program attempts to create a file on A. If it can't, the handler in the *criterr* unit gains control and reports the problem. The program then displays the error globals and quits. Because the error handler saves the screen before output, then restores it when it's finished, the error dialog simply vanishes.

The *Uses* statement at the top of Listing Two illustrates how units get linked with programs in Turbo Pascal 4.0. This program uses three units, the first two furnished with Turbo Pascal and the last the critical error handler from Listing One. Anything in the *Interface* section of a unit is accessible to the using program: constants, types, variables, and subroutines. Consequently, although *cerrtest* doesn't declare the *CriticalError* variables or define the *InstallCEH* procedure, it has access to them via the interface to *criterr*.

You might wish to make your own critical error handler less user dependent. In that case, remove all the I/O from *criterr* and simply load the reporting variables; your program can then decide what to do and how much to tell the user. By building in a custom critical error handler, you'll make your software much more bulletproof.

### *Who Am I?*

Because there's a new name on this column and it's mine, perhaps I should introduce myself. I've been

• DataWindows major update available
• OS/2 versions




# Avoid extra steps.

You've got better things to do than repeat the same steps. Over . . . and over . . . and over. Up your productivity with Greenleaf Software.

With more than 70 new functions added to our popular libraries, Greenleaf is now the most complete and mature C language function resource available. It's no wonder we've been rated the best. Winning program developers in major corporations such as IBM, EDS and GM have proven our reliability in thousands of applications.

*Step Lively*

New Greenleaf Functions v.3.10 includes 295 of the functions you've been asking for — DOS, disk, video, color text and graphics, string, time/date, keyboard, plus many more! With Greenleaf, you'll finish faster.

*Cut Corners*

When it comes to merging information, the new Greenleaf Comm Library v.2.10 is the fastest communications facility of its kind. Over 120 functions — ring buffered, interrupt-driven asynchronous communications. And, only Greenleaf gives you the power to build a 16-port communication system.

*Get on the Fast Track*

Order your new Greenleaf library today! See your dealer or call 1-800-523-9830.

| | |
|---|---|
| Greenleaf Comm Library | $185.00 |
| Greenleaf Functions | $185.00 |
| Greenleaf DataWindows | $225.00 |
| Greenleaf C Sampler | $ 94.50 |
| Digiboard Comm4 | $325.00 |
| Digiboard Comm8 | $535.00 |

In stock, shipped next day.

*Greenleaf DataWindows and Turbo C*

DataWindows, the finest C programming windows tool available, puts windows, transaction data entry and menus at your fingertips.

Our new TURBO C versions are ready to get you going fast! And our new 3-in-1 C Sampler for only $94.50 supports Turbo C or Quick C with comm, windows, menus and more! Our libraries support all popular C compilers for MS DOS.

GREENLEAF Software

Call Toll Free:
800-523-9830

In Texas and Alaska:
214-248-2561

Greenleaf Software, Inc.
16479 Dallas Parkway, Suite 570
Dallas, Texas 75248

CIRCLE NO. 167 ON READER SERVICE CARD

*(continued from page 104)*

hanging around computer rooms since the early 1960s doing various software and management jobs, and I was one of the micro pioneers when I got an IMSAI 8080 back in the Dark Ages, which was nine years ago. My first book was *Computers Made Really Simple,* published in 1976 and long out of print. Since then, I have written more than a dozen more, the most recent being *Stretching Turbo Pascal* (he mentioned with shameless commercialism). I've written another Pascal book as well, and I'm working on my second C book at the moment. I also write a lot of magazine articles, not only about programming but also about the application of small computers to business problems.

What I intend to do in this column is to present programming solutions—this installment being an example—as well as to review structured programming products and discuss issues relevant to software development.

But this isn't just my column, it's ours. If there's something you'd like to see here, drop me a line at *DDJ*. Or you can leave a note for KPORTER on MCI. Don't call, though, please; I don't work at the editorial offices. No promises, but I'll consider any reasonable suggestion. Right now I'm one guy in a room by himself, trying to guess what The Reader (that's you) wants. I won't know until you tell me. Let's make this into a forum that's fun and instructive for us both.

### *Availability*

All the source code for articles in this issue is available on a single disk. To order, send $14.95 to *Dr. Dobb's Journal,* 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

**DDJ**

**(Listings begin on page 84.)**

Vote for your favorite feature/article.
Circle Reader Service No. 2.


## *Programmer's Paradise Gives You Superb Selection, Personal Service and Unbeatable Prices!*

**Welcome to Paradise. The microcomputer software source that caters to your programming needs. Discover the Many Advantages of Paradise...**

- Lowest price guaranteed
- Latest versions
- Huge inventory, immediate shipment
- Knowledgeable sales staff
- Special orders
- 30-day money-back guarantee

***Over 500 brand-name products in stock — if you don't see it, call!***

## We'll Match Any Nationally Advertised Price.

| | LIST | OURS |
|---|---|---|
| **386 SOFTWARE** | | |
| ADVANTAGE 386 C OR PASCAL | 895 | 799 |
| MICROPORT SYSTEM V/386 (COMPLETE) | 799 | 679 |
| MS WINDOWS/386 *SPECIAL, NEW* | 195 | 119 |
| PHARLAP 386\|ASM/LINK | 495 | 419 |
| PHARLAP 386 DEBUG | 195 | 155 |
| SCO XENIX SYS V 386 (COMPLETE) | 1495 | 1195 |
| VM/386 *SPECIAL* | 195 | 119 |
| X-AM | 595 | 535 |
| **ARTIFICIAL INTELLIGENCE** | | |
| ARITY STANDARD PROLOG | 95 | 79 |
| MULISP-87 INTERPRETER | 300 | 199 |
| PC SCHEME | 95 | 85 |
| SMALLTALK/V *NEW* V. 2.0 | 99 | 85 |
| TURBO PROLOG | 100 | 65 |
| TURBO PROLOG TOOLBOX | 100 | 65 |
| **ASSEMBLERS/LINKERS** | | |
| ADVANTAGE DISASSEMBLER | 295 | 269 |
| ADVANTAGE LINK | 395 | 359 |
| ASMLIB | 149 | 125 |
| EZ_ASM | 70 | 65 |
| MS MACRO ASSEMBLER | 150 | 95 |
| PASM86 | 195 | 109 |
| PLINK86PLUS | 495 | 275 |
| RELMS, UNIWARE X-ASMS | CALL | CALL |
| VISIBLE COMPUTER 80286 | 100 | 89 |
| **BASIC** | | |
| DB/LIB | 139 | 119 |
| FLASH-UP | 89 | 79 |
| MACH 2 | 75 | 59 |
| MS QUICKBASIC | 99 | 65 |
| QUICKPAK | 69 | 59 |
| TRUE BASIC | 100 | 69 |
| TURBO BASIC | 100 | 65 |
| DATABASE TOOLBOX | 100 | 65 |
| EDITOR TOOLBOX | 100 | 65 |
| TELECOM TOOLBOX | 100 | 65 |

### XENIX/UNIX PRODUCTS

| | | |
|---|---|---|
| MICROPORT & SCO PRODUCTS | CALL | CALL |
| ADVANTAGE C++ | 695 | CALL |
| BTRIEVE | 595 | 455 |
| C-TERP | 498 | 379 |
| INFORMIX ESQL/C | 749 | CALL |
| INFORMIX 4GL | 1500 | CALL |
| INFORMIX SQL | 995 | CALL |
| KORN SHELL | 125 | 115 |
| MICROSOFT LANGUAGES | CALL | CALL |
| PANEL | 625 | 535 |
| PANEL PLUS | 795 | 675 |
| REAL TOOLS | 149 | 89 |
| RM/COBOL | 1250 | 949 |
| RM/FORTRAN | 750 | 549 |
| SCO MULTIVIEW (286) | 395 | 319 |
| SCO MULTIVIEW (386) | 495 | 399 |

| | LIST | OURS |
|---|---|---|
| **C++** | | |
| ADVANTAGE C++ | 495 | 479 |
| PFORCE++ | 395 | 209 |
| **C COMPILERS** | | |
| C86PLUS | 497 | 375 |
| LATTICE C | 500 | 265 |
| MICROSOFT C | 450 | 269 |
| QUICK C *SPECIAL* | 99 | 59 |
| TURBO C | 100 | 65 |
| **C INTERPRETERS** | | |
| C-TERP | 298 | 219 |
| INSTANT C | 495 | 369 |
| RUN/C | 120 | 79 |
| RUN/C PROFESSIONAL | 250 | 155 |
| **C LIBRARIES** | | |
| BASIC_C | 175 | 129 |
| C ASYNCH MANAGER | 175 | 135 |
| C-FOOD SMORGASBORD | 150 | 95 |
| W/SOURCE CODE | 300 | 179 |
| C TOOLS PLUS/5.0 | 129 | 99 |
| C UTILITY LIBRARY | 185 | 119 |
| C-XPERT | 395 | 339 |
| ESSENTIAL COMMUNICATIONS | 185 | 119 |
| COMMUNICATIONS PLUS | 250 | 189 |
| GREENLEAF C SAMPLER | 95 | 69 |
| GREENLEAF COMM LIBRARY | 185 | 125 |
| GREENLEAF FUNCTIONS | 185 | 125 |
| MULTI-C | 149 | 135 |
| PFORCE *SPECIAL* | 295 | 199 |
| RESIDENT C W/SOURCE | 198 | 169 |
| TIMESLICER | 295 | 265 |
| W/SOURCE CODE | 1000 | 895 |
| TURBO C TOOLS | 129 | 99 |
| **COBOL** | | |
| E-Z PAGE | 295 | 259 |
| MICRO FOCUS | | |
| COBOL/2 *SPECIAL* | 900 | 729 |
| COBOL/2 TOOLSET *NEW* | 900 | 729 |
| PC-CICS | 1500 | CALL |
| LEVEL II COBOL | 349 | 279 |
| PERSONAL COBOL | 149 | 119 |
| OTHERS | CALL | CALL |
| MICROSOFT COBOL | 700 | 429 |
| MICROSOFT SORT | 195 | 129 |
| OPT-TECH SORT | 149 | 99 |
| REALCICS | 995 | 785 |
| REALIA COBOL | 995 | 785 |
| W/REALMENU | 1145 | 899 |
| RM/COBOL | 950 | 759 |
| RM/COBOL-85 | 1250 | 999 |
| RM/SCREENS | 395 | 315 |
| SCREENIO | 400 | 379 |
| **DEBUGGERS** | | |
| ADVANCED TRACE-86 | 175 | 115 |
| C-SPRITE | 175 | 119 |
| PERISCOPE I | 345 | 275 |
| PERISCOPE II | 175 | 139 |
| PERISCOPE III 8 MHZ | 995 | 799 |
| PERISCOPE III 10 MHZ | 1095 | 875 |
| PFIX 86 PLUS | 395 | 209 |
| T-DEBUG PLUS | 60 | 49 |
| XVIEW86 | 60 | 49 |
| **DISK/DOS/KEYBOARD UTILITIES** | | |
| COMMAND PLUS | 80 | 69 |
| DISK OPTIMIZER | 60 | 55 |
| FETCH *SPECIAL* | 55 | 45 |
| INTELLIGENT BACKUP | 150 | 135 |
| NORTON COMMANDER | 75 | 55 |
| ADVANCED NORTON UTILITIES | 150 | 99 |
| PDISK | 145 | 99 |
| VFEATURE | 80 | 75 |
| VFEATURE DELUXE | 120 | 110 |
| **EDITORS** | | |
| BRIEF | 195 | CALL |
| W/DBRIEF | 275 | CALL |
| EDIX | 195 | 155 |
| EMACS | 295 | 265 |
| EPSILON | 195 | 149 |
| KEDIT | 125 | 99 |
| PC/EDT | 250 | 229 |
| PC/VI | 149 | 109 |
| PLEDITOR *SPECIAL, NEW* | 195 | 149 |
| PMATE | 195 | 109 |
| SPF/PC | 195 | 145 |
| VEDIT PLUS | 185 | 129 |
| **FILE MANAGEMENT** | | |
| BTRIEVE | 245 | 185 |
| XTRIEVE | 245 | 185 |
| REPORT OPTION | 145 | 99 |
| BTRIEVE/N | 595 | 455 |
| XTRIEVE/N | 595 | 455 |
| REPORT OPTION/N | 345 | 269 |
| CBTREE | 159 | 139 |
| C-TREE | 395 | 315 |
| R-TREE | 295 | 239 |
| C-TREE/R-TREE BUNDLE | 650 | 519 |
| DBC III | 250 | 169 |
| DBC III PLUS | 750 | 595 |
| DB_VISTA OR DB_QUERY | 195 | 159 |
| SINGLE USER W/SOURCE CODE | 495 | 399 |
| MULTIUSER | 495 | 399 |
| MULTIUSER W/SOURCE CODE | 990 | 789 |
| INFORMIX PRODUCTS | CALL | CALL |
| PHACT MANAGER | 249 | 219 |
| XQL *NEW* | 795 | 599 |
| **FORTRAN COMPILERS** | | |
| LAHEY FORTRAN | 477 | CALL |
| LAHEY PERSONAL FORTRAN 77 | 95 | 89 |
| MICROSOFT FORTRAN | 450 | 269 |
| RM/FORTRAN | 595 | 479 |
| **FORTRAN UTILITIES/LIBRARIES** | | |
| DIAGRAM'ER OR DOCUMENT'ER | 129 | 115 |
| DIFF-E-Q | 495 | 445 |
| FORTRAN ADDENDA | 165 | 139 |
| GRAFMATIC OR PLOTMATIC | 135 | 119 |
| MAGUS NUMERICAL ANALYST | 295 | 249 |
| MATHPAC | 495 | 445 |
| NO LIMIT | 129 | 109 |
| SPINDRIFT LIBRARY | 149 | 135 |
| SSP/PC | 350 | 269 |
| **GRAPHICS** | | |
| ADVANTAGE GRAPHICS (C) | 250 | 225 |
| ESSENTIAL GRAPHICS | 250 | 185 |
| GSS GRAPHIC DEV. TOOLKIT | 495 | 375 |
| HALO *SPECIAL* | 300 | 199 |
| HALO (5 MICROSOFT LANG.) | 595 | 389 |
| METAWINDOW PLUS | 275 | 229 |
| TURBOWINDOW/C | 95 | 79 |
| TURBO HALO (FOR TURBO C) | 99 | 79 |
| **LINT** | | |
| PC-LINT | 139 | 99 |
| PRE-C | 295 | 155 |
| **MODULA-2** | | |
| LOGITECH MODULA-2 | | |
| COMPILER PACK | 99 | 79 |
| DEVELOPMENT SYSTEM | 249 | 199 |
| TOOLKIT | 169 | 139 |
| WINDOW PACKAGE | 49 | 39 |
| ROM PACKAGE AND CROSS RUNTIME DEBUGGER | 299 | 239 |
| REPERTOIRE | 89 | 75 |
| **TURBO PASCAL ADD-ONS** | | |
| DOS/BIOS & MOUSE TOOLS | 75 | 69 |
| FLASH-UP | 89 | 79 |
| METRABYTE DATA ACQ. TOOLS | 100 | 89 |
| SCREEN SCULPTOR | 125 | 89 |
| SYSTEM BUILDER | 150 | 129 |
| IMPEX | 100 | 89 |
| REPORT BUILDER | 130 | 115 |
| T-DEBUG PLUS | 60 | 49 |
| TURBO.ASM | 99 | 69 |
| TURBO ASYNCH PLUS | 129 | 99 |
| TURBO EXTENDER | 85 | 65 |
| TURBO HALO | 99 | 79 |
| TURBO MAGIC | 199 | 179 |
| TURBO OPTIMIZER | 75 | 65 |
| TURBO POWER TOOLS PLUS | 129 | 99 |
| TURBO POWER UTILITIES | 95 | 79 |
| TURBO PROFESSIONAL 4.0 *NEW* | 99 | 79 |
| TURBO WINDOW/PASCAL | 95 | 79 |

### FEATURED PRODUCTS

**MICROSOFT WINDOWS/386** — Software for 386-based machines offering true multi-tasking. Run several programs within different windows on your screen. Cut and paste between them. Each program is given 640K of memory.
List: $195 **Special Price: $119**

**MICRO FOCUS COBOL/2** — Compiler system that can exploit the full memory of 80286/386 machines under OS/2 or under PC-DOS. Conforms to the highest certifiable level of ANSI'85 COBOL. Includes the ANIMATOR source code debugger and offers full network support.
List: $900 **Special Price: $729**

**PANEL/QC or TC** — This best selling screen management library is now available for both Quick C and Turbo C. Supports pop-up fields and windows, multi-line fields, horizontal and vertical field scrolling, menus, help boxes, and custom field validation. Generates C source code. No royalties.
List: $129 **Special Price: $89**

**FETCH** — New memory resident file librarian, for those of us who have trouble finding files, or forget what's in them. The user is prompted for a 255 character description anytime a file is created. Later, Fetch can scan the file description library with its fast pattern recognition capabilities, and will display the description, directory, and drive.
List: $55 **Special Price: $45**

| | LIST | OURS |
|---|---|---|
| **OPERATING SYSTEMS** | | |
| MICROPORT SYSTEM V/AT | 549 | 465 |
| SCO XENIX SYSTEM V | 1295 | 989 |
| WENDIN-DOS | 99 | 79 |
| OTHER MICROPORT, SCO, WENDIN PRODUCTS | CALL | CALL |
| **PASCAL COMPILERS** | | |
| MARSHAL PASCAL | 189 | 155 |
| MICROSOFT PASCAL | 300 | 185 |
| PASCAL-2 | 350 | 319 |
| TURBO PASCAL *NEW* V. 4.0 | 100 | 65 |
| TURBO PASCAL DEV.LIB. *NEW* | 395 | 259 |
| BORLAND ADD-ONS | CALL | CALL |
| **SCREEN DISPLAY/WINDOWS** | | |
| C-SCAPE | 279 | 265 |
| CURSES W/SOURCE CODE | 250 | 169 |
| GREENLEAF DATA WINDOWS | 225 | 155 |
| W/SOURCE CODE | 395 | 259 |
| HI-SCREEN XL | 149 | 119 |
| JYACC FORMAKER | 495 | 449 |
| JYACC JAM | 750 | 679 |
| MICROSOFT WINDOWS | 99 | 65 |
| MS WINDOWS DEVELOPMENT KIT | 500 | 309 |
| PANEL PLUS | 495 | 395 |
| PANEL/QC OR TC *SPECIAL* | 129 | 89 |
| QUICKSCREEN | 195 | 175 |
| SCREENSTAR W/SOURCE | 198 | 169 |
| VITAMIN C | 225 | 149 |
| VC SCREEN | 99 | 79 |
| VIEW MANAGER | 275 | 199 |
| WINDOWS FOR DATA *SPECIAL* | 295 | 229 |

### HARDWARE PRODUCTS

| | | |
|---|---|---|
| AMDEK 722 MONITOR | 750 | 499 |
| AMDEK 730 MONITOR | 899 | 569 |
| AST ADVANTAGE PREMIUM W/512K | 495 | 319 |
| AST RAMPAGE! 286 W/512K | 545 | 349 |
| HERCULES GRAPHICS CARD PLUS | 299 | 195 |
| HERCULES IN COLOR CARD | 499 | 329 |
| IRMA 2 | 1195 | 779 |
| ORCHID TURBO EGA | 749 | 495 |
| ORCHID TURBO PGA | 1495 | 1099 |
| PARADISE SYSTEMS | | |
| VGA PROFESSIONAL | 599 | 409 |
| AUTOSWITCH EGA 480 CARD | 349 | 169 |
| VEGA DELUXE | 379 | 259 |
| **ADDITIONAL PRODUCTS** | | |
| ADVANTAGE VCMS | 379 | 329 |
| BASTOC | 495 | 399 |
| CARBON COPY PLUS | 195 | 159 |
| DAN BRICKLIN'S DEMO PROGRAM | 75 | 59 |
| DB2C | 299 | CALL |
| FLOW CHARTING II | 229 | 205 |
| MAGIC PC | 195 | 179 |
| MKS TOOLKIT | 139 | 115 |
| NORTON GUIDES | 100 | 65 |
| PFINISH | 395 | 209 |
| POLYMAKE | 149 | 125 |
| POLYTRON PVCS | CALL | CALL |
| SOURCE PRINT | 95 | 75 |
| TREE DIAGRAMMER | 77 | 69 |

**Terms and Policies**
- **We honor MC, VISA, AMERICAN EXPRESS**

No surcharge on credit card or C.O.D. Prepayment by check. New York State residents add applicable sales tax. Shipping and handling $3.00 per item, sent UPS ground. Rush service available, prevailing rates.
- Programmer's Paradise will match any current nationally advertised price for the products listed in this ad.
- Prices and Policies subject to change without notice.
- Hours 9AM EST — 7PM EST
- *Ask for details. Some manufacturers will not allow returns once disk seals are broken.

**Corporate Buyers — Call for special discounts and benefits!**

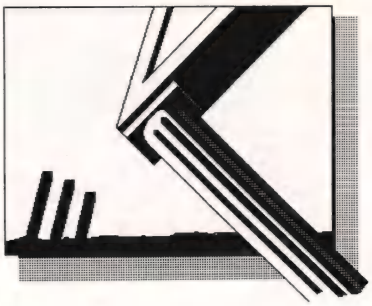
**1-800-445-7899**
**In NY: 914-332-4548**
**Customer Service: 914-332-0869**
**International Orders: 914-332-4548**
**Telex: 510-601-7602**

Programmer's Paradise™
A Division of Hudson Technologies, Inc.
42 River Street, Tarrytown, NY 10591

CIRCLE NO. 170 ON READER SERVICE CARD

# THE FORTH COLUMN

## Update on Forth in the Industry



This last year has been a busy one for Forth. I like to believe that the "Forth year" starts and ends with FORML (Forth Modification Laboratory) at Asilomar in Northern California over the Thanksgiving weekend. I'll give you a complete report of this year's FORML (87) in the next column. Meanwhile, here's what's happened since the last FORML (86). I apologize in advance for missing anything.

The Novix 4000 (renumbered to 4016) continued rising in popularity. Both Novix and Software Composers released Novix boards for the IBM PC. The Novix NB4100 contains the NC4016 with B and X ports wired to board-mounted connectors and 128K of program and data memory. The Software Composer PC4000 has 500K of memory on-board and can run in parallel with other PC4000 boards. Each company offers Forth development systems as well as C-to-Forth translation programs. Novix also announced the NB4300 STD-bus Novix card. (An STE-bus Novix card is available from Forth Systeme, W. Germany.)

Harris Semiconductor announced FORCE (Forth-optimized RISC computing engine), a modified NC4016 design copied into its macro cell library. It expects to offer a FORCE-based, real-time control processor (RTCP) in the first quarter of 1988.

Phil Koopman demonstrated his WISC (writable instruction set, stack-oriented computer) CPU/32, a modular hardware Forth engine sold by

by Martin Tracy

Mountain View Press. The team from Johns Hopkins University demonstrated its Forth engine, a 32-bit processor with cached stacks and Gbytes of address space. The team expects it to be generally available sometime in 1988.

Meanwhile, Forth continued to move into newer and more challenging hardware environments. Dr. C.H. Ting implemented a Novix-based micro-code sequencer for NCR's GAPP computer. Goddard Space Lab (Maryland) implemented Forth on its massively parallel processor, a two-dimensional array of 128 × 128 serial processors. FORTH Inc. announced a polyFORTH for the Texas Instruments TMS32020/C25 digital signal processor (DSP). Forth is currently the only high-level language running directly on this popular DSP chip.

FORTH Inc. also implemented a nicely optimized Intel 80386 polyFORTH running in native mode. The company reports that Forth benchmarks run faster on this machine than they do on the Motorola 68020. Meanwhile, Laboratory Microsystems demonstrated its new UR/FORTH running on the 80286 and 80386 under Microsoft's OS/2.

Forth vendors were busy last year applying their tools to a variety of projects. Miller Microcomputer Services put the finishing touches on MMS Forth 2.4, the MS-DOS version of Forth used to develop RapidFile, Ashton-Tate's new query-by-example database. The new book *Business Programming with RapidFile* was written by our own Leo Brodie, whose improved *Starting Forth*, second edition, was also published last year.

Creative Solutions turned its talents inward to develop new hardware technology. The result was a series of boards for Apple's Macintosh II NuBus. Three of these Hurdler boards connect the Mac II to three other buses: the STD bus, the Motorola I/O channel, and the IBM PC bus. CSI also announced the Mac II Toolkit, which adds a 68020 assembler, 6881 support, and color graphics as extensions to its popular MacFORTH.

FORTH Inc. spent a busy year working with several *Fortune* 500 companies. The company assisted in developing a package-tracking magic wand for a major mailing company and renovated the American Airlines baggage system, a polyFORTH system that has been in place for the past five years. It expanded Bell Canada's data entry system (70–80 users on one VAX VMS with no loss of response) and demonstrated a large factory heating ventilation and air-conditioning system using the ClusterFORTH distributed intelligence network.

Advance MicroMotion developed an economy Telex and EasyLink communications package for TTS. Mountain View Press brought out new MVP Forths for the Amiga, the Atari 600/800/1200, and the PDP-11. New Micros announced a hardware and software development system for the Motorola 68HC11, and Inner Access Corp. announced its development system for the Zilog SUPER8 chip.

Besides RapidFile, two other major Forth products appeared. Electronic Arts' STARFLIGHT game took several programmer-years of effort (see *Forth Dimensions*, July 1987). Frog Peak Music (P.O. Box 9911, Oakland, CA 94613) brought out the Hierarchical Music Specification Language, which lets you write your own synthesizer MIDI driver and edit waveforms, durations, and envelopes as you play.

Last year also saw the birth of a new Forth computer, sort of. The Canon Cat (*Byte*, October 1987) is Jef Raskin's first new machine since he left Apple, where he headed the original Macintosh team. The Cat has a

FORTH COLUMN
*(continued from page 108)*

9-inch mono display, 3.5-inch drive, and a Motorola 68000 CPU. According to Ezra Shapiro, "If the highlighted text [on the screen] is a computer program written in either Forth or 68000 assembly language, the Cat executes it."

And of course, the ANSI CBEMA X3J14 Forth standardization effort began last year. I'll report on the second meeting of this committee in the next column. Also, there are two new Forth electronic bulletin boards: the FIG-sponsored GENie board (see *DDJ*, December 1987) and the new North Coast Forth Board (Minnesota [612] 483-6711). Now there are Forth boards north, east, and west. South Coast Forth Board anyone?

### *Forth in AI*

Forth continued to make inroads into artificial intelligence. Henry Harris (JPL) presented papers on conceptual dependency; William Dress (Oak Ridge National Lab) presented several on neural nets. Two implementations of OPS5 and two of "fuzzy" rule production systems were developed last year. Jack Park's popular Expert 2 system evolved to Expert 5. For a general summary of Forth's recent history in AI, see "The Forth Wave in AI" by Robert Trelease in *AI Expert* (October 1987).

Part of Forth's popularity in AI is the ease in which it can be implemented on experimental computer architectures. This makes Forth particularly attractive for simulating, controlling, and testing inference engines and neural networks. Once Forth is ported to a new hardware environment, it is possible to implement additional languages, such as BASIC, LISP, and PROLOG, rapidly by writing them in Forth. Panasonic, for example, implemented BASIC this way on the original HHC (handheld computer). Charles Duff's

```
/STRING ( a n n2 - a+n2 n-n2)
\ truncates leftmost n chars of string.
\ n may be negative.

EVAL ( a n )
\ evaluates ("text interprets") a string.

ASCII ( - c)
\ returns value of following char.

CTO"" ( c - a 1)
\ converts character to string.

SKIP ( a 1 c - a2 12)
\ returns shorter string from
\ first position unequal to byte.

SCAN ( a 1 byte - a2 12)
\ returns shorter string from
\ first position equal to byte.

" ( - a )
\ STATE-smart string literal.

VAL? ( a n - d 2 , n2 1 , 0)
\ string to number conversion.
\ True if d is valid.
\ Returns d if number contains ",-./:"
\ and sets DPL = 0
\ Returns n if no punctuation present
\ and sets DPL = 0<

VAL ( a n - d f)
\ converts string to double number.
\ True if number is valid.

-TEXT ( a n a2 - -1 , 0 , 1)
\ returns -1 if string a n < a2 n ,
\ 0 if equal, and 1 if >.

COMPARE ( a n a2 n2 - -1 , 0 , 1)
\ returns -1 if a n < a2 n2 , \ 0 if equal, and 1 if >.

MATCH ( a n a2 n2 - ???? 0 , offset -1)
\ returns the position of
\ string a2 n2 in (a n).
\ Offset is zero if ( a n )
\ is found in first char position.
\ False with invalid offset
\ if ( a n ) isn't in a2 n2.
```

***Example 1:*** *String operators*

object-oriented NEON and Actor were both written as extensions to Forth, too.

By the way, a tiny Modula-2 has just been written in Forth by S. Lohr. You can download it from the East Coast Forth Board ([703] 442-8695) or from the new GENie Forth Board as the file TM2.ARC. You will find an interesting discussion of language bootstrapping techniques in "Embeddings of Languages in Forth" by R.D. Dixon in the latest *Journal of Forth Application and Research*, vol. 4, no. 4, 1987.

This same issue contains two important articles on implementing PROLOG in Forth: "A Full PROLOG Interpreter Embedded in Forth" by Odette and Paloski and "Compiling PROLOG to Forth" by Odette. The latter article contains careful and expert notes on implementing the compiler, with complete source code. Compiling PROLOG to run on Forth hardware gives you an inference engine with truly awesome speed. Lou Odette reports that a "naive reverse benchmark" runs at 6,000 LIPS (logical inferences per second) on an NC4016 with a (conservative) 4-KHz clock. When the next generation of Forth processors appears later this year, they should run the same benchmark at the speed of compiled (Quintus) PROLOG on a VAX 11/780.

Flash! At this year's annual Forth Convention (San Jose, Calif., November 1987) Silicon Composers unveiled an IBM PC AT board containing the FORCE core set from Harris Semiconductor. FORCE is implemented on the board as five separate chips: the Forth-based CPU, a hardware multiplier, an interrupt controller, a data stack, and a return stack. All pertinent signals are bused to a prototyping area on the board. It has 32K of high-speed RAM, expandable to 128K (as soon as high-density 35-nsec RAM chips become available).

The development system includes an optimizing compiler that reads standard ASCII text files. You could use this system to prototype and test hardware macros before integrating them into a custom VLSI Forth-based RISC machine. The AT/FORCE board is available from Silicon Composers ([415] 322-8763) starting mid-December 1987 and retails for $4,500.

### Text and Data Files

In my last column (December 1987), I looked at a minimal but useful set of string operators, which are summarized in Example 1, page 110. The word *MATCH* replaces the word *–MATCH* in the last column. Its source code is in this month's listing (see Listing One, page 86). –MATCH is the name of a polyFORTH word that compares strings cell by cell and is used primarily for searching index files.

The string package was built on the Standard prelude (see *DDJ*, October 1987) and the *DDJ* Controlled Reference Word Set:

```
2* D2* HEX C,
BL ERASE BLANK .R
2>R 2R> AGAIN DLITERAL
S>D WITHIN TRUE
```

Definitions for these words are also in the listing.

This month's topic is accessing files from Forth. You may have heard that Forth is both a language and an operating system. It's true. Because Forth is often the first development system to work in a new environment, it needs to manage memory, handle interrupts, and access mass storage. The essential primitives for reading and writing mass storage are *BLOCK* and *UPDATE*. Eventually, when other operating systems become available, Forth may be extended to coexist peacefully with them.

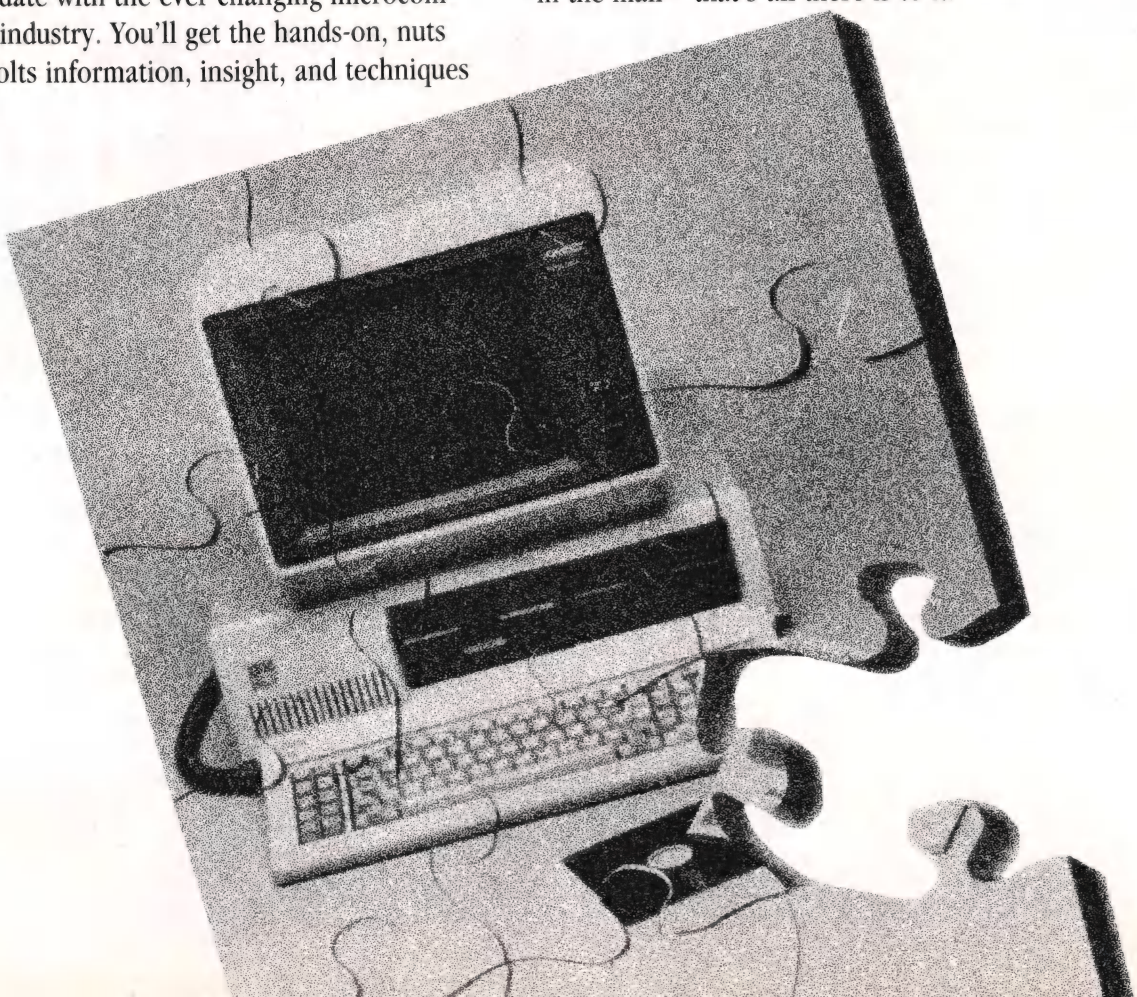
When Forth is itself the operating system, it divides mass storage into consecutively numbered (1,024-byte) blocks. Both source code and data are kept in these blocks. There are no "text" or "data" files in the normal sense. Any source or data read by Forth is previously written by Forth.

When Forth runs under an operating system, it takes advantage of operating system calls. Under CP/M, for example, any logical sector in a file can be read or written directly. Sectors can be grouped into blocks, and so any file can be accessed as a sequence of blocks.

But what if the file is not an integer number of blocks long? The file

# Why puzzle when you don't have to?

*Micro/Systems Journal* has the answers. Whether it's networking, systems integration, programming, or scientific computing questions, *M/SJ* will lead you out of the maze of microcomputer mayhem. With each issue you'll find comprehensive coverage of all the technical information that will keep you up-to-date with the ever-changing microcomputer industry. You'll get the hands-on, nuts and bolts information, insight, and techniques that *M/SJ* is famous for providing . . . in-depth tutorials, reviews, hints, the latest on multitasking, languages and operating systems. So stop your puzzling . . . subscribe right now and the answers will be yours. Simply drop the attached card in the mail—that's all there is to it.



# MICRO/SYSTEMS

## JOURNAL FOR THE PC SYSTEMS INTEGRATOR

## SUBSCRIBE NOW AND SAVE OVER 47% OFF THE NEWSSTAND PRICE!

47% SAVINGS

Yes! I want to subscribe to

MicroSystems

JOURNAL for the PC Systems Integrator

**and save over 47% off the cover price.**

☐ 1 Year (12 issues) $29.97 ☐ 2 Years (24 issues) $56.97

**Please charge my:** ☐ Visa ☐ Master Card ☐ American Express

☐ Payment Enclosed ☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

*Savings based on the full one-year newsstand rate of $47.70. All foreign countries please add $7 per year for surface mail; Canada & Mexico add $17 per year for airmail; other countries add $28 per year for airmail. All foreign subscriptions must be paid in U.S. funds drawn on a U.S. bank. Please allow 6-8 weeks for delivery.

A PUBLICATION OF M&T PUBLISHING, INC. 210S

47% SAVINGS

Yes! I want to subscribe to

MicroSystems

JOURNAL for the PC Systems Integrator

**and save over 47% off the cover price.**

☐ 1 Year (12 issues) $29.97 ☐ 2 Years (24 issues) $56.97

**Please charge my:** ☐ Visa ☐ Master Card ☐ American Express

☐ Payment Enclosed ☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

*Savings based on the full one-year newsstand rate of $47.70. All foreign countries please add $7 per year for surface mail; Canada & Mexico add $17 per year for airmail; other countries add $28 per year for airmail. All foreign subscriptions must be paid in U.S. funds drawn on a U.S. bank. Please allow 6-8 weeks for delivery.

A PUBLICATION OF M&T PUBLISHING, INC. 210S

47% SAVINGS

Yes! I want to subscribe to

MicroSystems

JOURNAL for the PC Systems Integrator

**and save over 47% off the cover price.**

☐ 1 Year (12 issues) $29.97 ☐ 2 Years (24 issues) $56.97

**Please charge my:** ☐ Visa ☐ Master Card ☐ American Express

☐ Payment Enclosed ☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

*Savings based on the full one-year newsstand rate of $47.70. All foreign countries please add $7 per year for surface mail; Canada & Mexico add $17 per year for airmail; other countries add $28 per year for airmail. All foreign subscriptions must be paid in U.S. funds drawn on a U.S. bank. Please allow 6-8 weeks for delivery.

A PUBLICATION OF M&T PUBLISHING, INC. 210S

No Postage
Necessary
If Mailed
In The
United States

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

Micro Systems
JOURNAL for the PC Systems Integrator

Box 3713
Escondido, CA 92025-9843

No Postage
Necessary
If Mailed
In The
United States

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

Micro Systems
JOURNAL for the PC Systems Integrator

Box 3713
Escondidio, CA 92025-9843

No Postage
Necessary
If Mailed
In The
United States

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

Micro Systems
JOURNAL for the PC Systems Integrator

Box 3713
Escondidio, CA 92025-9843

# EVEN MORE POWER AND FLEXIBILITY

# BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now, the best gets better, with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features. Many users find that BRIEF is the only editor they'll ever need, with features like real, multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF into your own custom editor containing the commands and features you desire. It's fast and easy.

Jerry Pournelle, columnist for BYTE magazine summed it all up by saying BRIEF is, "**Recommended. If you need a general purpose PC programming editor, look no further.**" His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFOWORLD AND PC MAGAZINE.

One user stated "BRIEF is one of the few pieces of software that I would dare call a masterpiece." Order BRIEF now and find out why. **BRIEF 2.0 is just $195. If you already own BRIEF, call for upgrade information.**

**TO ORDER CALL: 1-800-821-2492 (in MA call 617-337-6963)**

As always, BRIEF comes with a 30 day money-back satisfaction guarantee.

Solution Systems™

541 Main Street
Suite 410D
So. Weymouth, MA 02190
(617) 337-6963

## Look at these BRIEF 2.0 enhancements!

### Main Features:

- All new documentation with tutorials on basic editing, regular expressions **and** the BRIEF Macro Language.
- Setup program for easy installation and configuration. (Requires no knowledge of the macro language)
- Increased speed for sophisticated operations like Undo and Regular Expression Search.
- Expanded regular expressions, with matching over line boundaries.
- More block types, with marking by character, line or column.
- Command line editing (move cursor, add and delete characters, specify command parameters).
- Support for more programming languages.
- Optional borderless windows.
- Enhanced large display support, including wider displays.
- Reconfigurable indenting for C files (supports most indenting styles).

Plus the basic features that made BRIEF **SO** popular!

### Basic Features:

- Full multi-level Undo
- Windows
- Edit many files at once
- File size limited only by disk space
- Automatic language sensitive indentation


Requires an IBM PC or compatible with at least 192K RAM.
BRIEF is a trademark of UnderWare, Inc.
Solution Systems is a trademark of Solution Systems.

CIRCLE NO. 174 ON READER SERVICE CARD

*(continued from page 111)*

will only fill part of the last block. This leads to the first rule of standard file access from Forth:

1. *BLOCK* must be able to read the entire file. If the last block is partial, the contents of *BLOCK* past the end of the file are undefined. Reading a partial block is not an error condition.

What about writing to a partial block? The UPDATE command tells Forth only that some part of a block has been written to but not which part. This leads to a second rule:

2. UPDATE forces a partial block to be extended to a full block. The file length must be adjusted accordingly.

If you use Forth only to read Forth files, you will see no partial blocks and neither rule will apply. Otherwise, two operating system calls are required:

1. to determine the length of a file in some unit easily convertible to a double number of bytes
2. to extend the length of a file by a multiple of the same unit

The easiest way to keep track of the length of a file is to call the system when the file is first opened and to maintain its double-number length in a *2VARIABLE* (or equivalent)—for example, *CAPACITY*. Furthermore, assume that you can construct the word *EXTEND*, which extends the file by at least a given double number of bytes. Suppose, for example, that Forth provides the word *MORE*, which extends a file by a given number of blocks:

```
: MORE ( n ) . . . ;
: EXTEND ( d )
\ extends file d bytes.
   1024 UM/MOD SWAP
   IF 1+ THEN (round up)
   MORE ;
```

In the interests of keeping primitives primitive, assume that whoever calls *EXTEND* also maintains *CAPACITY*. Some operating systems extend files automatically, so you can use a null definition for *EXTEND*:

```
: EXTEND ( d ) COMPILE 2DROP ;
\ null definition.
   IMMEDIATE
```

*BLOCK* and *UPDATE* give you direct access to a file. Data stream and text files, however, use sequential access, reading and writing the next group of characters or bytes. This implies that you should maintain a current (double-number) byte position for the file, which can be kept in a *2VARIABLE* (or equivalent)—call it *POSITION*. *POSITION* is initialized to 0 when the file is first opened. The words *GETDATA* and *PUTDATA* can now be defined to read or write the next *n* bytes of a file to or from a memory location:

```
: GETDATA ( a n – n2) . . . ;
   \ reads n bytes of data from file
   \ into address, n < 64K. Returns
   \ n2 bytes not read, ie beyond
   \ end of file.
   : PUTDATA ( a n) . . . ;
   \ writes n bytes of data to file
   \ from address, n < 64K.
```

High-level definitions of these words are in Listing One. If possible, you should implement them as system calls instead.

Given *POSITION*, *CAPACITY*, *GETDATA*, and *PUTDATA*, you can immediately implement *GETLINE* and *PUTLINE* to access text files. A text file is simply a sequence of text lines, which are in turn a sequence of characters terminated by an end-of-line condition or an end-of-file condition. These conditions are typically implemented as special characters. Each line may be terminated by an optional line-feed character:

```
: GETLINE ( a n – a n2 f) . . . . ;
\ reads n bytes of text from file
\ into address, n < 64K. n2 bytes
\ are actually read. True if end-
\ of-line terminates line.

: PUTLINE ( a n ) . . . ;
\ writes n bytes of data to file
\ from address, n < 64K.
```

*GETLINE* is based on *GETTEXT*, which is based on *GETDATA*. The end-of-line flag is needed because either end-of-line or end-of file can return a zero-length string.

Note that all the words I've just described can ultimately be built from *BLOCK* and *UPDATE*, provided that the given rules and assumptions are valid. Nothing has been said about file selection, which changes greatly from Forth to Forth. Selecting the proper file before accessing it is up to you. Most operating-system-oriented Forths allow you to select one of at least two files, typically one for input only and one for output or modify. When you change files, you must update or switch *CAPACITY* and *POSITION* accordingly.

Now that you can access text files, you can do some truly wonderful things:

```
: TYPE-FILE
\ reads and prints a file.
   BEGIN PAD 80 GETLINE
   WHILE CR TYPE
   REPEAT 2DROP ;
```

It really is that easy. In the listing, I have included examples for copying text files and for converting blocks to text and back again. One final example shows you how to interpret text files, which you can create using Borland's SideKick or some other handy text editor:

```
: EVAL-FILE
\ reads and interprets a file.
   BEGIN PAD 80 GETLINE
   WHILE EVAL
   REPEAT 2DROP ;
```

### *Availability*

All the source code for articles in this issue is available on a single disk. To order, send $14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

**DDJ**

**(Listing begins on page 86.)**

# DEBUGGING SWAT TEAM

*Order Eco-C88 Rel. 4.0 New Modeling Compiler and get C-more at no extra charge!*

## Seek and Correct

You already know that fast compilation does not mean fast program development. Backing up for bogus error messages and removing the bugs takes time. Eco-C88's "Seek and Correct" three - way error checking finds even the most elusive bugs, clearing the path for swift program development.

## Double Barrel Error Checking

Eco-C88 nails **syntax errors** cold and tells you about the error in plain English. And there's no avalanche of false error messages, either. Other compilers can generate up to four times the number of error messages actually present; they leave it up to you to guess which ones are real. You'll be more productive with Eco-C88 because there is no guess work.

Eco-C88 provides ten levels of **semantic error** checking. You can select from almost no checking to the fussiest you've ever seen. Eco-C88's "picky flag" finds subtle errors that slip by other compilers.

## Eco-C88 also features:

- All data types, plus ANSI Enhancements
- Robust library, including many new ANSI functions
- CED editor with online function help, split windows, compile-edit-link capability
- New, expanded manual with sample programs for the library functions

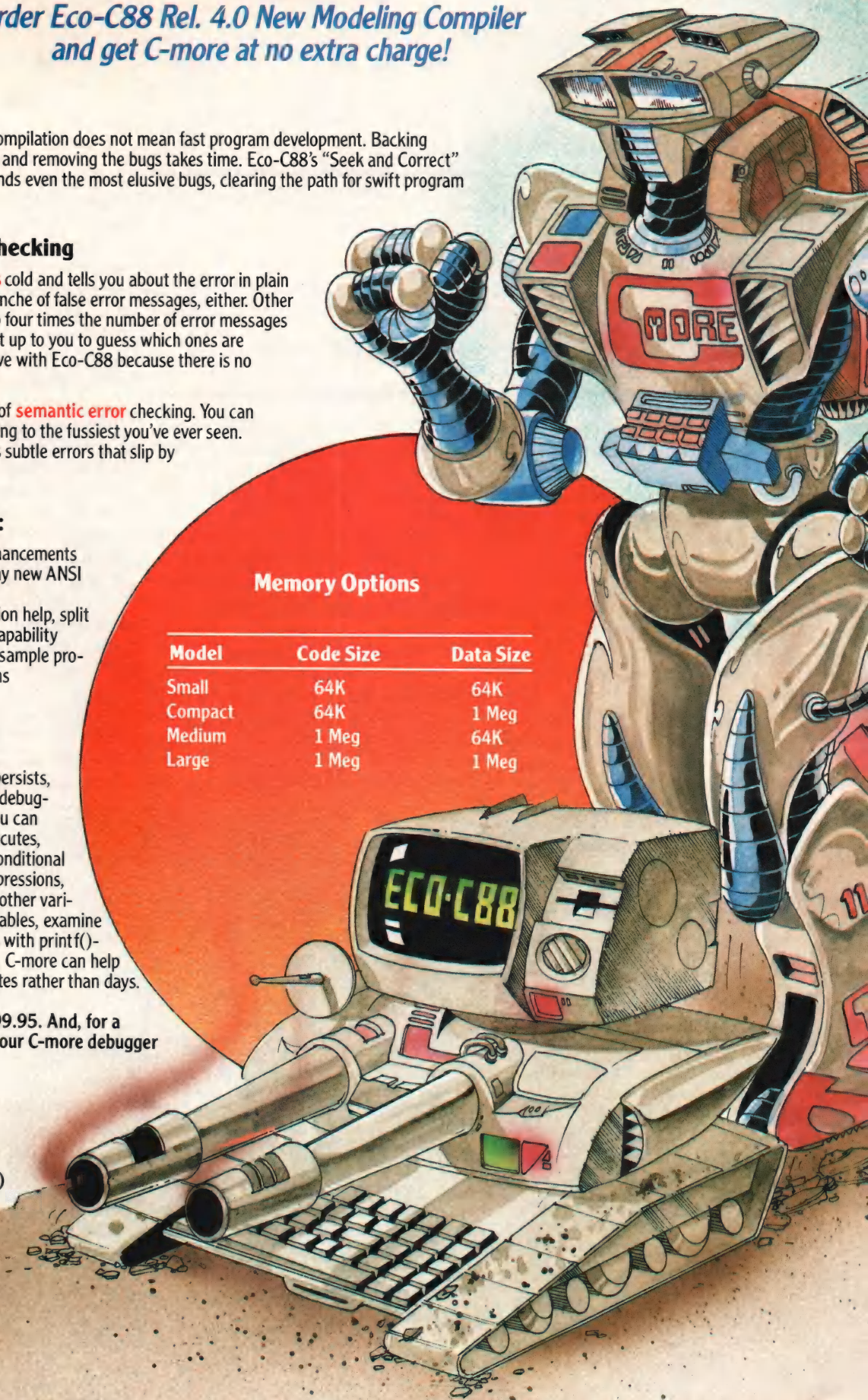
## C-more Source Code Debugger

Finally, if a really nasty bug persists, put C-more, our source code debugger, to work. With C-more you can watch your program as it executes, single-step it, set simple or conditional breakpoints, test complex expressions, use variables as indexes into other variables, initialize and trace variables, examine CPU registers, display results with printf()-type options and much more. C-more can help you track down bugs in minutes rather than days.

**The price for Eco-C88 is $99.95. And, for a limited time, we'll give you our C-more debugger at no extra charge.**

### Memory Options

| Model | Code Size | Data Size |
|---|---|---|
| Small | 64K | 64K |
| Compact | 64K | 1 Meg |
| Medium | 1 Meg | 64K |
| Large | 1 Meg | 1 Meg |




**Ecosoft Inc.**
6413 N. College Ave.
Indianapolis, IN 46220

(317) 255-6476 (Tech Info)
(800) 952-0472 (Orders)

CIRCLE NO. 175 ON READER SERVICE CARD

# EXAMINING ROOM

# The Norton Guides

**Target:**
*PC or PS/2 and compatibles*
**Requires:**
*DOS 2.0 or later. Hard disk recommended*
**Pricing:**
*$50 for the program, $50 for each language guide*
**Vendor:**
*Peter Norton Computing Inc. 2210 Wilshire Blvd, Santa Monica 90403 (213) 453-2361*

Some years ago, after watching me work, my daughter announced to her playmates that a programmer is someone who sits in front of a computer looking things up in books. It seemed funny at the time, but the more I do it, the more I realize she was right. For the uninitiated, here's how to program: open to the index of manual A, turn to half a dozen references before you find what you need, make two keystrokes, then open manual B and repeat the process. While compiler-makers toil mightily to shave milliseconds off compile time, we plod through weighty tomes checking error codes, parameter-passing conventions, syntax diagrams and all the rest. Simply put, the greatest obstacle to programmer productivity is the everlasting need to look things up.

Well, Peter Norton has done something about it with his Norton's Guides. The glib hype says it's a product for those who hate manual labor. A more specific description is that it's on-line, language-specific help for serious programmers. Yes, it's another memory-stealing 37K Terminate-and-Stay-Resident program, but think how much desk space it clears up.

Any time you're in text mode—editing a program, using a debugger, whatever—Shift-F1 wakes up the Norton Guides by giving you a half-screen pop-up containing a menu bar. Thoughtfully, the pop-up appears in the half away from the cursor, so that it doesn't obscure the code you're working on. You can toggle to full-screen with F9, and make the pop-up go away with Esc or F10.

And what do you get with the pop-up? Initially, a list of every instruction in the language on a scrollable panel. You can also do a keyword search that takes you straight to the instruction you want. Positioning the highlight over an instruction, you press Enter and *shazam!* you get a detailed description of its purpose and syntax, often with a code example. There are also cross-references to related statements and topics, to which you can jump by pointing to a pick list entry and pressing Enter.

Additionally, the Guides furnish information about specific features of a given language implementation (the library functions of Turbo C and Microsoft C, for example, as well as data types, reserved words, etc.). Another menu selection called Tables leads to all those little details that consume so much look-up time in manuals: the ASCII values of line-drawing and graphics characters, color codes, etc. In short, the Norton Guides furnish, with a few keystrokes, access to most of the information programmers need while in a fever of creativity.

The foundation of the Guides is a $50 program that makes it work. Then you'll also need one or more of the language databases, each priced at $50. The entry price, then, is $100, plus $50 per language thereafter. If you have more than one database, the Options menu lets you switch among them with a couple

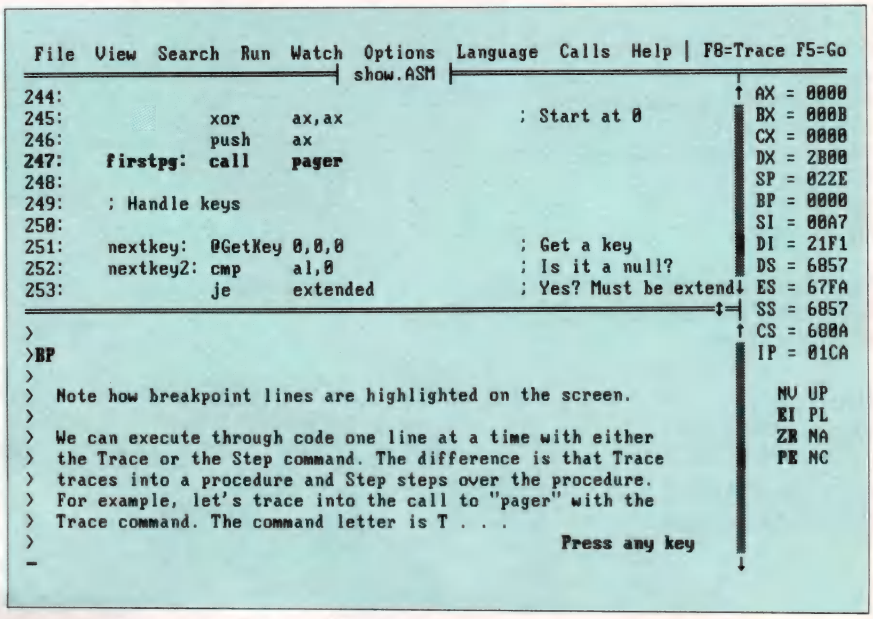



***Figure 1:*** *The Norton Guides add on-line help to a number of compilers, including Turbo C.*

# 10 REASONS WHY YOUR ADA COMPILER SHOULD BE A JANUS/ADA COMPILER (Besides the Price!)

1. **We're fast**

Compilation and development speed mean a lot when you're working on a large program; every minute you have to wait is time and money wasted. A Janus/Ada compiler has an average compilation speed of 400 lines per minute with a 6 MHZ 8086; over 500 lines per minute with an 80186; 1100+ lines per minute on an 8 MHZ 80286, and nearly 1900 lines per minute on PC Limited's 386.

2. **We work on any XXX86 machine.**

Not everybody can afford the newest technology and some of us can't afford the name brands . . . but with Janus/Ada, that doesn't matter. It doesn't need protected mode, 8087's, or PC-DOS. As long as you have an Intel 80X86 chip at the heart of your computer and some way to run or emulate MS-DOS 2.1 or higher, you can use Janus/Ada.

3. **Our product has been in use for over six years.**

We released our first compiler in 1981 and got our first review in 1982. The products we sell have had the scrutiny of customers and reviewers for over six years, changing to meet the increased sophistication of the technology and the user. Our customers come from every segment of the computing community; the government, Fortune 500, and academia. Users like yourself are able to benefit from the wide range of uses we respond to every day, as well as the built-in network of people working with Ada just like you.

4. **No copy protection.**

Our products come to you free of any form of copy protection and with a warranty insuring media replacement for defective media. We know you don't need the hassle of locking devices and other protective devices. We know you want to use Janus/Ada on your hard disk and network, and we make it easy to install, easy to maintain, and easy to use.

5. **Application programs written in Ada for as low as $12.00.**

We offer IBM graphic routines, window handlers, scientific libraries and many other application programs written in Ada. These programs come with complete source code and documentation and are user tested.

6. **We offer embedded systems capability, like rommable, reentrant code and licensing of our source code.**

Janus/Ada is designed to make your embedded processing easy . . . with features especially created for that environment. Our code is rommable, allowing you to "burn in" the applications you create. We make our source code available, in our Systems Package ("S" Pak), so you don't have to guess at the answer. If you would prefer to have us design and implement your code, we'll be more than happy to accommodate.

7. **Superior error handling.**

Nothing is more frustrating than an error in coding . . . except a bad error message which does nothing except confuse you. Our error messages tell you vital information, such as line location, cause of error and a full walkback including subprogram names. If you do a lot of development, or you just appreciate being fully informed, our error messages are sure to please!

8. **Support that won't let you down.**

Superior support services, such as the Janus/Ada bulletin board, the Janus/Ada BIX conference, a quarterly newsletter, as well as customized support services.

9. **Our compiler was developed on a micro, for a micro, in Ada.**

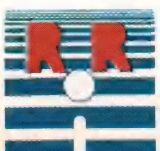
We have "bootstrapped" our compiler, version by version, so that we know it really works! Our compiler was developed on multiple XXX86 machines, from different manufacturers, so that it would be a true microcomputer compiler. Our products are developed in Ada, demonstrating our commitment to the language and the capabilities it can bring to you.

10. **Self paced, low cost educational tutorials.**

The Janus/Ada Extended Tutorial offers a programmer's perspective on learning Ada. This tutorial was written expressly for novices in the Ada language, by a programmer, and includes a complete set of disk based quizzes to compliment the 150 page tutorial. You can determine how quickly you pace yourself, and you won't believe you did it for $99.95.

**JANUS/ADA PASSED VALIDATION 12/11/87 . . . AND IS STILL ONLY $99**





SOFTWARE, INC.

*specialists in state of the art programming*

P.O. Box 1512 Madison, Wisconsin 53701
(608) 244-6436 TELEX 4998168

1-800-722-3248

CIRCLE NO. 176 ON READER SERVICE CARD

*(continued from page 116)*

of keystrokes.

As of this writing, Guides are available for BASICA, QuickBASIC, Turbo BASIC, Turbo Pascal, Turbo C, Microsoft C, and MASM. A spokeswoman for the company says more are in the works and should be out by the time you read this: the OS/2 kernel and Microsoft Windows are two that she'd 'fess up to.

For serious programmers, I recommend a set of two databases: MASM and the high-level language you normally use. That's because the Assembler Guide contains a wealth of DOS-related information not found in the others: error codes, interrupts, file attributes, and descriptions of the PSP and FCB structures, among others. If you want to reach deep into the machine from C or Pascal, you need this stuff.

There's a mistake in the Tables, which are common to all incarnations of the Guides. It says the screen attribute byte is given by 256 * background + foreground; replace 256 with 16. I reported it to the vendor, so it should be fixed in future releases. The Turbo C sample program for vprintf() closes a nonexistent stream. Other than that, my only complaint is that the tutorials are cloyingly cutesy. These minor points are enormously outweighed by the convenience of having a comprehensive language reference instantly available at one's fingertips.

In addition to the basic TSR, the program package includes tools for rolling your own guides. Norton calls them a compiler and linker, but don't be misled; they constitute a text formatter governed by a half-dozen commands embedded in the ASCII files. These "bang commands" associate text lines with menus, set up cross-references, establish linkages among text files, and so on. Once you've linked your files, they become a database available through the Guides' Options menu. It takes about 15 minutes to master the whole process.

This opens the way to tremendous possibilities for application developers. Instead of laboriously constructing help screens and all the code required to manage them, you

# Some of the world's biggest problems are being solved with a touch of Smalltalk.


## On the ground floor of high-tech environmental control.

*Climate, energy, fire and security are all critical aspects of environmental control in large office buildings. The challenge for Johnson Controls, a leader in this industry, is to provide a control system that is both technologically advanced and simple to operate. Using Smalltalk/V, Research Scientists Gene Korienek and Tom Wrensch have developed a prototype that integrates all environmental functions into a real-time control system using sophisticated color graphics on a PC. This enables the building engineer to immediately spot and correct problems by simply clicking a mouse. And provides building owners and tenants with a better climate for business.*

## Abroad and involved in foreign affairs.

*The French Ministry of Foreign Affairs is responsible for keeping track of every French citizen living abroad and every foreigner living in France. Each day, they process thousands of requests for documents or information, each one of which takes at least fifteen minutes. Arthur Andersen, the world's largest accounting firm, has developed a natural language processing application with Smalltalk/V that enables clerks without computer training to extract the necessary data much faster. Thanks to Smalltalk and system developers Bart Schutle and Pascal Wattiaux, what once took fifteen minutes now takes 30 seconds. Vive la Smalltalk!*

The world is made of objects. So naturally, the world is turning to Object-Oriented Programming (OOPS). And the fastest, easiest OOPS language and environment is Smalltalk/V.

With OOPS you program by defining objects, their inter-relationships and their behavior. Objects can represent both real-world entities—people, places, things—as well as useful abstractions such as stacks, sets and rectangles. Smalltalk/V provides everything you need to solve problems big and small, including a comprehensive tutorial to get you started.

### Who needs Smalltalk?

Because Smalltalk models the way people really think, it is perfect for scientists, engineers and professionals who have to solve tough problems in a short amount of time. Perfect for programmers who are looking for a fast, efficient prototyping environment. And anyone who wants to quickly and easily learn OOPS.

### Introducing a bigger and better OOPS: Smalltalk/V286.

Our newest version of Smalltalk offers faster and more powerful OOPS capabilities We've gone from 16 to 32-bit architecture.


### Teaching students to think economically.

*With OOPS and Smalltalk, even non-programmers can create exciting applications. Economics Professor Arnold Katz of the University of Pittsburgh developed Economics PC Discovery World, an intelligent tutoring system for his beginning microeconomics students. Using a mouse to access a series of windows and manipulate data, a student can call up a set of markets and commodities for an imaginary community. By changing the scenario, the student can not only study a variety of market behaviors, but also test the validity of his or her own reasoning. A process that provides a lot of food for thought.*

From 640K to 16 MB capacity for 25 times the memory. And designed it to run on the next generation OS/2 operating system as well as DOS.

### Get Smalltalk for a small price.

Smalltalk/V sells for just $99.95. Smalltalk/V286 is $199.95. The following optional applications packs are available for $49.95 each: Communications; EGA/VGA Color; Goodies #1; Goodies #2, Carleton Tools and Goodies #3, Carleton Projects.

And everything comes with a 60-day, money-back guarantee.

So visit your nearest dealer. Or call toll-free, 800-922-8255 and order direct with MasterCard or Visa. Or write to Digitalk, Inc., 9841 Airport Blvd., Los Angeles, CA 90045.

And let us help you put Smalltalk into action.

# Smalltalk/V

digitalk inc.


Smalltalk/V requires DOS and 512K RAM on IBM PC/AT/PS or compatibles and a CGA, MCGA, EGA, VGA, Toshiba T3100, Hercules, or AT&T 6300 graphic controller. A Microsoft or compatible mouse is recommended. Not copy protected. Smalltalk/V286 requires a 286 or 386, DOS or OS/2 and 1MB of memory and one of the graphic controllers list above.

CIRCLE NO. 178 ON READER SERVICE CARD

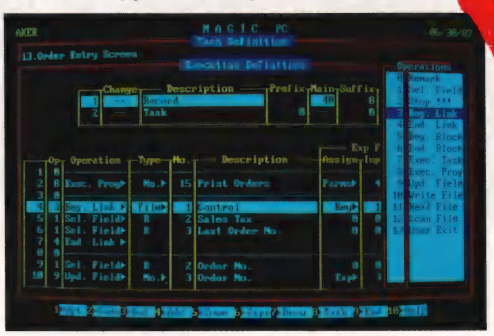
# MAGIC PC: A REVOLUTION IN POWER, PRICE & PROGRAMMING SPEED.

**You know how database applications are created — by hacking out line after line of time-consuming code. Most DBMS' and 4GL's give you some programming power. But when it comes to serious applications, they keep you bolted to your seat writing mountains of tedious code. And rewriting it all over again with every design change.**

**Imagine how much faster you'd be if you could replace the painful coding phase with an innovative visual technology which takes only a fraction of the time: Introducing Magic PC—the revolutionary Visual Database Language from Aker Corporation:**

## High-Speed Programming:

With Magic PC's visual design language you quickly describe your programs in non-procedural Execution Tables. They contain compact programming operations which are executed by Magic PC's runtime engine. You fill-in the tables using a visual interface driven by windows and point-and-shoot menus. One table with 50 operations eliminates writing more than 500 traditional lines of code. Yet with Magic PC you don't sacrifice any power or flexibility.



*With a powerful set of high-level non-procedural operations you program at only a fraction of the time.*

## Maximum Power AND Simplicity:

With Magic PC, you can generate robust DBMS applications including screens, windows, menus, reports, forms, import/export, and much more! Plus, Magic PC has one of the friendliest user interfaces you've ever seen. Using Magic PC you can look-up and transfer data through a powerful Zoom Window system. Magic PC even lets you perform command-free queries.

## Btrieve Performance:

Magic PC incorporates Btrieve, the high-performance file manager from SoftCraft. This gives you exceptional access speed, extended data dictionary capabilities, and automatic file recovery!

## Virtually Maintenance-Free:

With Magic PC you can modify your application design "on the fly" without any manual maintenance. Magic PC automatically updates your programs and data files on-line! This also makes Magic PC an ideal tool for prototyping complete applications in hours instead of days.

## FREE Networking:

Magic PC comes complete with LAN features. Develop multi-user applications for your LAN with Magic's file and record-locking security levels.

## Stand-Alone Runtime:

Distribute your applications and protect your design with Magic PC's low cost runtime engine.

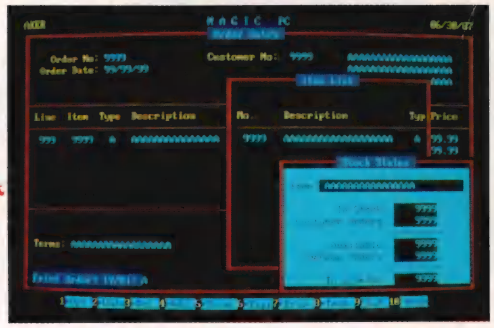
## All For Only $199:

Best of all, Magic PC is an unbeatable bargain. For a limited time, Magic PC's price has been reduced to only $199! Yes, this is the same Magic PC that normally lists for $695! And Magic PC eliminates the need for a separate DBMS, compiler, or application generator. It comes complete with all the tools you need to develop your own database applications instantly.


## $199 – With A Money-Back Guarantee!

For a limited time, you can get Magic PC for only $199. And even at this low price, Magic PC is risk-free. If you're not completely satisfied, simply return it within 30 days and we'll buy it back (less $19.95 restocking fee). And if you'd like a preview, Magic PC's Tutorial Demo is available for just $19.95.

But you'd better hurry – Magic PC's special $199 price won't last long!



*Pop-up Zoom Windows run multiple programs per screen — with point-and-shoot data transfer between windows!*

## Join The Magic PC Revolution

To unleash your DBMS design power, order your $199 copy of Magic PC right now by calling toll-free or returning the coupon below.

## ORDER NOW: CALL (800) 345-MAGIC In CA (714) 250-1718

*"Magic PC's data base engine delivers powerful applications in a fraction of the time. . . there is truly no competitive product."*

Victor Wright – PC Tech Journal

Also recommended by: *PC Magazine, PC World, PC Week, Computer Language, Data Base Advisor,* and many other publications worldwide.

**MAGIC PC**
The Visual Database Language
by AKER


Yes! I want to generate powerful applications much faster!

☐ Rush me my copy of Magic PC at the special promotional price of $199 (add $10 P&H, and tax in CA. International orders add $30). I understand I can return Magic PC for a refund within 30 days, if I'm not completely satisfied.*

☐ Rush me a copy of Magic PC Tutorial Demo at $19.95 (add $5 P&H, and tax in CA. International orders add $15).

Name _____

Company _____

Street Address (no POB) _____

City _____ State _____ Zip _____

☐ Check enclosed Charge to my: ☐ VISA ☐ MasterCard ☐ AMERICAN EXPRESS

Account No. _____

Acct. Name _____ Exp. Date ____/____

Signature _____

Return to: **Aker Corp.**, 18007 Skypark Cir B2, Irvine, CA 92714

DD

System requirements: IBM PC, XT, AT, PS/2 or 100% compatible with 512K RAM, hard disk and DOS 2.0 or later. 5¼" format, not copy protected. Dealer pricing available. *Return policy valid in US only.

*(continued from page 118)*

can write a Guides database and bundle it with your software. Norton Computing says they're willing to license distribution rights to the underlying TSR program.

The Norton Guides don't completely eliminate the need for manuals and reference books, but they reduce clutter around the keyboard. They also let you write code a whole bunch faster, which is good news. This product represents a quantum jump in programmer productivity.

*by Kent Porter*

# Soft-Scope Source-Level Debugger

**Target:**
*PC/XT and compatibles*
**Required:**
*DOS 2.0 or later, hard disk, 256K. Compiler/linker generating Intel-type symbols*
**Price:**
*(DOS version only, others available) Debugger is $500 One-year update/ support is $100*
**Vendor:**
*Concurrent Sciences Inc. P.O. Box 9666 Moscow, ID 83843; (208) 882-0445*

Throw out the quiche, put the dog in the back of the pickup, and crack open a beer; Soft-Scope is a real man's debugger. No wimpy windows, no frilly function keys, no cuddly colors. No Sir, this is purely a command-driven program on a monochrome screen that keeps on scrollin'. It might be argued that real men don't use symbolic debuggers at all, but those who do will appreciate the full features and solid feel that underlie Soft-Scope's rugged exterior.

Soft-Scope reminds us of the forgotten fact that software doesn't have to be cute to be useful. What it lacks in sex appeal, this debugger makes up for in functionality. It monitors and reports program execution at the source level in any of Pascal, C, FORTRAN, Assembler, and even PLM and JOVIAL. You can set breakpoints on source lines and refer to variables by name when checking or assigning values. If you want to see what the compiler generated from a source line, you can view the resulting machine language as assembly code.

The point of reference when working with Soft-Scope is the source line number, which it gets from the requisite .LST file. Like a BASIC interpreter, it tracks its position in the source file via line numbers, and lists ranges of lines on command. It also associates a line number with a subroutine name, so that you can refer either by line or by name to entry points for setting breaks and the like. A command lets you find out the range of lines for a given routine. This is useful for setting a watchpoint within that range only, using the commands "RANGE 100 120" and "GO TIL $X = 5$."

The single-step trace mode works by default at the source line level. You get into single-step with the S command, then keep punching the space bar to execute successive lines. Soft-Scope displays each line as it executes it. If you get tired of this, you can hit Enter instead of the space bar, then specify a speed from 0 (slowest) to 9 (fastest), and the debugger runs in continuous mode, displaying each line as it runs. The Q key (or alternatively Esc) stops it so that you can examine variables. You can also disassemble the source line and single-step at machine-code level, which is a nice feature for uncovering those bizarre glitches that sometimes crop up in programs that tickle the deep innards of the machine.

Soft-Scope has the particularly attractive ability to display the nesting of calls by name. A program crashes in some subroutine and you wonder how on earth it got there; this tells you who called whom, back up to the main program level. You can also list the stack itself to any level from the top down; e.g. "20 STACK"

lists the top 20 words. Another command lists the stack size and the high-water mark, handy for diagnosing crashes due to stack overrun.

This debugger understands aggregate data structures. You can display a structure by name, listing its fields and data types, and fetch any of the fields' values. The same is true of arrays, which can be dumped as a whole, by row, or by individual element.

One of the best features of Soft-Scope is pointer dereferencing. This allows you to follow an indirection chain by dereferencing a series of pointers to an object, whose value appears on the screen. You can also view the resolved address to find out if you've got a screwy pointer. The pointer dereferencing notation is pure Pascal, which might confound those without a Pascal background. Still, it's not difficult to master. If INTPTR is a pointer to an integer, then the value of the integer is represented by INTPTR^. Similarly, STRUCT^.FLD^.OBJ is an indirection chain in which STRUCT points to FLD, which points to OBJ.

Another feature worth mentioning, is Soft-Scope's ability to route all of its prompts and displays to an external RS-232 terminal. This feature makes the program well suited for debugging programs which are highly interactive with the screen cursor position or are graphical in nature.

Soft-Scope's primitive user interface is usually not a problem. Indeed, elaborate displays such as CodeView's often dazzle the eye and thus get in the way. It would be nice, though, if you didn't have to halt the trace in Soft-Scope and type a command to examine registers and variables. Also, the inability to switch display pages in Soft-Scope means that program and debugger output get mixed together, which can lead to a wild jumble.

Soft-Scope has 37 commands, most of which have several optional modifiers. The program acknowledges human frailty to the extent of furnishing a HELP command that accepts an argument. For example, "HELP SUBMIT" furnishes a synopsis of the command syntax for exercising a debugger script file.

But until you're a certified Soft-Scope guru, you can't effectively work this debugger without the manual at your elbow. The documentation is readable if a bit terse. There are 305 pages of tutorials using several programming languages, and the command reference is 98 pages. A major gripe with the manual is that it lacks an index.

Soft-Scope comes in several flavors. The one I reviewed is for MS-DOS. There are also incarnations for the likes of RTCS/UDI and iRMX. Prices range from $500 for the DOS version up to $2000 for the Intel 310 with iRMX 286 OS, and considerably higher than that for versions working with the Applied Microsystems ES 1800 emulator. Clearly, this is a debugger for serious developers.

Despite its no-frills, macho interface, Soft-Scope delivers for those who can afford it.

*by Kent Porter*

*(continued on page 124)*


# THE NEW 386 MOTHERBOARD.

# THE ANSWER TO YOUR PC'S PRAYERS.

## 386 SPEED—ONLY $1,495

Don't let your PC give up the ghost—Hauppauge has just arrived with a new spark of life: the 386 MotherBoard. Our new MotherBoard graces your PC, PC/XT or compatible with speeds equal to the IBM PS2 Model 80. And faster. Because we've built in 1 Megabyte of high speed RAM and a 387 math coprocessor socket for speeds that make even irreverent users humble with awe.

**OS/2 Compatible** To ensure a long, fruitful life, our 386 MotherBoard is compatible with the PC/AT (BIOS and I/O)— allowing you to run the new generation of DOS, OS/2. Equally inspiring, our MotherBoard runs Windows/386, UNIX V and PC-MOS/386. And if you need more power, you'll find two 16-bit expansion slots that accommodate the latest I/O expansion card. No 386 accelerator card gives you so much versatility. **Only our 386 MotherBoard gives your PC a future with unlimited possibilities!**

**Technical Features** ▪ 16 MHz 80386 ▪ 1 Megabyte of 100 nsec 4-way interleaved RAM ▪ PC/AT compatible I/O and BIOS for support of OS/2 ▪ Six 8-bit expansion slots ▪ Two 16-bit expansion slots ▪ One 32-bit expansion slot for up to 12 Megabytes of high speed memory ▪ Battery-powered clock calendar ▪ Optional 16 MHz 80387 math coprocessor ($695).

*"Judged on price, performance, and ease of installation, the Hauppauge 386 MotherBoard is easily the Upgrade Product of the Year."*
—Robert Luhn, PC World

For more information on our easy-to-install MotherBoard, call: **1 (800) 443-6284**. In New York, call **(516) 434-1600**.

Hauppauge Computer Works, Inc.
175 Commerce Drive,
Hauppauge, New York 11788

Hauppauge!

CIRCLE NO. 181 ON READER SERVICE CARD

*(continued from page 123)*

# XO-Shell

**Target:**
*PC or PS/2 and compatibles*
**Requires:**
*DOS 2.0 or later*
**Pricing:**
*$49*
**Vendor:**
*Wyte Corp.*
*701 Concord Ave,*
*Cambridge, MA 02138; (617) 868-7704*

Every once in a while you run across a smoothly crafted little program and you just hate the guys who dreamed it up: Why couldn't I have thought of that? The only consolation is that it's probably almost as much fun to use as it was to write it.

XO-Shell is such a program. I'm not sure what to call it. It's not really a debugger, nor is it exactly a DOS shell. The vendor refers to it as a programmer's productivity aid. Okay, but it could be a productivity tool for non-programmers, too. Agreement on one point is assured: XO-Shell is slick. It does a lot of stuff that you wish DOS and/or your editor would do.

I tend to have an allergic reaction to TSR's (Terminate-and-Stay-Resident programs). They can so overload memory with conveniences that you don't have enough left to do real work. And so I viewed XO-Shell with a certain wariness, especially in view of its 88K requirement. To exact that kind of price, I figured, it'd better be good.

Your jaundiced reviewer was pleasantly surprised. There's a whole host of goodies that make that 88K worth the cost. On the more prosaic side, XO-Shell stores up DOS commands for recall, edit, and re-execution, and it has a file viewer similar to Xtree and other DOS shells. What sets XO-Shell apart from other shells is that you can do this stuff from within an application such as an editor or the Turbo/Quick environments. But that's not all.

Say you're editing a program module and you've forgotten the exact spelling of a variable defined in a different module. With XO-Shell, you can pop up the other module and check it, or print out a portion of that code, without leaving your edit session. Whoops! You just noticed an error in the other module. No sweat, XO-Shell lets you correct it on the spot, then escape back to your main editing job. All it takes is the Alt-n hot key and a few keystrokes.

Last month you wrote a nifty little algorithm, and now you need to plug it into your latest masterpiece. Position the cursor where you want it to go, bring up that old file with XO-Shell, and mark the text with some Wordstar-like commands. When you leave XO-Shell, it copies the marked text into the new program, effecting a clipboard.

Maybe you remember the name of that algorithm, but not the name of the source file containing it. XO-Shell has a grep-like facility to search file groups for a specific string. Function keys direct the search results to the screen, the printer, or a file,

"Multiple windows."

"I can run a shell or DCL command session even while editing."

"Programming modes to assist me in writing C, Pascal, ADA and Lisp."

"I can associate any command to any keystroke sequence."

"Electronic mail is only one of the many included packages I use."

"Emacs runs on MS-DOS, VMS and UNIX. It's my company's standard editor."

"A hundred times better than the UNIX vi editor."

"UniPress really supports Emacs on Berkeley and AT&T UNIX, and is constantly developing Emacs."

"Emacs automates the code-compile-edit-debug process."

"I can create my own customized editor commands with the MLisp extension language."

# No Two Users Can Agree On Why UNIPRESS EMACS™ Is The Only Choice.

## *Now Supports X Windows and Suntools Pop-up Menus On Any Terminal*

That's right. No two users can agree on why they've chosen UniPress Emacs. UniPress Emacs provides so much for creating and modifying programs, documents and other text files. No other program has such powerful facilities. And we offer Emacs for UNIX,™ Xenix,™ Ultrix,™ MS-DOS™ and VMS™ systems.

UniPress Emacs gives you all the basic text editing commands, and goes much further. It allows multiple windows to display several files or portions of the same file simultaneously, a program's output, a shell/DCL window, a help window, an error listing, an electronic mail message, or anything else!

UniPress Emacs simplifies programming with its C, Pascal, ADA and Lisp modes. It checks for balanced parentheses and braces, and indents and reformats code as needed. "C mode" produces templates of control flow in user-definable styles. After compiling, Emacs collects any compiler error output in a window. Emacs then analyzes this error output and automatically displays the erroneous source lines, one-by-one.

In addition, Emacs offers great extensibility through macros and the built-in compiled MLisp™ programming language. The MLisp language provides looping, conditional testing and other flow control statements, and a rich set of logic, arithmetic and string operations which can be combined.

If you're still not convinced about UniPress Emacs, call us at **800-222-0550** (Outside NJ) to find out about our **dial-in demo** machine or send in the coupon to get more product information. We'll send you our **free catalog** with information about Emacs and our other UNIX software products.

**UniPress Software, Inc.**, 2025 Lincoln Highway, Edison, NJ 08817. Telephone: 201-985-8000. Order Desk: 800- 222-0550 (Outside NJ). Telex: 709418

UniPress Software

Trademarks of: UniPress Emacs & MLisp, UniPress Software, Inc.; UNIX, AT&T Bell Laboratories; VMS, & Ultrix, Digital Equipment Corp.; MS-DOS, Xenix, Microsoft.

CIRCLE NO. 183 ON READER SERVICE CARD

EXAMINING ROOM
*(continued from page 124)*

and you can save the files list for future use in wide-ranging searches. XO-Shell also searches the file currently being worked on, in case your editor lacks this elementary capability.

Few things are as annoying as having to leave an edit session and list the directory simply because you forgot the name of an include file or data file. XO-Shell lets you list a directory while editing. It can also search directories to find a specific file if you can't remember which sub it's in. And if, while browsing among files, you decide it's time to clean up some directories, XO-Shell has the ability to copy and erase. When you're done, hit Esc and resume editing.

So now the program is done, but there's a variable name you want to change, or maybe one that's declared but possibly not used. Pop into XO-Shell and do a cross-reference listing among the program modules. It lists every line, identified by file, where the variable occurs.

There are other things as well. For example, XO-Shell can save output screens from your program in a disk file, and restore them later for double-checking. For another, there's a pop-up via Alt-m that shows all 256 characters; pick one from the display and plug it into your edit screen. Even if your editor can't generate graphics characters, you can incorporate them into the text this way. And XO-Shell lets you drag them around and replicate them, providing for text boxes and other visual effects in source listings, as well as direct encoding of special characters in string literals.

Aside from the hoggish 88K, my only criticism of XO-Shell is that the helpful hints on the prompt line (which appears only in response to the hot key) aren't very helpful. Such things as "F9:Chng_para" are far from intuitive, even for C cowboys.

XO-Shell has won its way into my programming toolkit, and I recommend it.

*by Kent Porter*

*(Examining Room continued on page 130.)*

# Our front end helps protect your back end.

Today's users require sophisticated interfaces for their applications. Yet complex front ends are a real pain to create, especially when the specs change and your deadlines don't.

But now JYACC introduces JAM™, a powerful user interface development tool that makes it easier than ever to design and revise your complex applications.

*Use windows and colors freely with JAM.*

JAM is the first tool that does it all, from prototyping to implementation. With JAM, you start by creating screens and linking them together to develop an application shell. You can experiment with the look of the interface, and even explore "what if" scenarios on the application flow. Then you attach processing routines, and your application is complete.

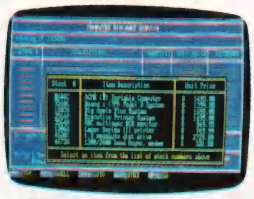
JAM works under the following operating systems:

- UNIX®
- MS-DOS®
- VMS®
- XENIX®
- RMX™
- VOS™

You'll be amazed at how quickly your applications spring to life with JAM's interactive screen management utility. You can use features like context-sensitive help, shifting and scrolling fields, a variety of visual attributes and extensive data validations to create exciting screens, windows and menus—all without writing a single line of code. JAM also lets you test your prototypes at any time.

JAM lets you draw from its extensive subroutine library to help you write processing routines faster. And revisions to your applications are easier, because your subroutines are insulated from the data entry and presentation details of the interface.

JAM is extremely portable. It runs on almost every computer, from PCs to superminis, and works under 6 operating systems. This allows you to develop consistent interfaces throughout your company—a significant asset to the Fortune 500 companies that have been using JAM for more than a year.



*Enhance applications with context-sensitive help.*

JAM from JYACC. It gets your front ends—and back ends—in great shape. Call for more information about JAM and our demo diskette. 800-458-3313

JYACC FORMAKER™, JAM's screen manager, is also available separately.

JYACC, Inc., 116 John Street
New York, NY 10038 212-267-7722

Introducing
# JAM

JYACC Application Manager. *The Composer for Sophisticated Applications.*

JYACC

Excellence in Systems & Application Design

MS-DOS, XENIX: Microsoft Corp.; UNIX: AT&T Bell Labs; RMX: Intel Corp.; VMS: Digital Equipment Corp.; VOS: Stratus Corp.

**CIRCLE NO. 185 ON READER SERVICE CARD**

# NEW! TOOLKITS FOR TURBO C & QUICK C from ZORTECH INC.


## HOTKEY

*A complete set of Terminate Stay Resident (TSR) functions that help you to write reliable 'pop-up' programs.*

*Now you can make your programs 'Sidekickable'. Two example programs are included, a 'pop-up Calculator' and a pop-up 'Critical Error Handler'.*

*The Hotkey toolkit handles all floating point functions in resident mode.*

*The 32 page manual includes an interesting discussion of the origin and history of undocumented MS-DOS function calls, together with a full explanation of the theory and practical use of TSR's.*

**Only $49.95!** State Turbo C or Quick C version.)


## COMMS

*Do you need to incorporate serial communications into your applications? Yes! Then get this inexpensive but highly professional COMMS toolkit from Zortech Inc.*

*Look at the list of features: Xmodem, Kermit and ASCII file transfer, Hayes modem control, VT52, VT100 and ANSI terminal emulation, supports up to 8 serial ports, speeds up to 19.2k baud rate and higher.*

*Two demonstration programs are included, MINICOM and MAXICOM (like Procomm) together with the 120 page manual and full source code FREE!*

**Only $49.95!** (State Turbo C or Quick C version.)



## GAMES

*Have you ever wondered how to write a chess program? Now we reveal the secret algorithms and techniques of the masters with this dynamic Games toolkit.*

*The package comes complete with the full source code to three ready to play games of strategy – Chess, Backgammon and Wari (an ancient African game).*

*A comprehensive 150 page manual is provided giving an in depth look at the history, structure and program design of such 'Strategy Games'.*

**Only $49.95!**

(State Turbo C or Quick C version.)



Turbo C is the trademark of Borland Inc. Quick C is the trademark of MicroSoft Inc.

## SUPERTEXT

*This is not simply an 'Editor' toolkit, but a full-blown, 'WordStar' compatible wordprocessor with the full source code.*

*As well as all the normal editing functions, you will also find 'dot' commands and full printer control. The SuperText toolkit handles files of any size and allows full on-screen configuration.*

*Do you need to incorporate a wordprocessor into your application? Yes! Then get the SuperText toolkit complete with full source code and 150 page manual now!*

**Only $49.95!** (State Turbo C or Quick C version.)

## PROSCREEN

*Generate high quality data entry screens with the Pro-Screen – Screen Designer and Code Generator.*

*You can draw the data entry screen, define the input fields, define the input criteria, set screen colors and attributes, draw single or double lines, make boxes – press a few buttons and 'hey presto' Pro-Screen generates the C source code for your application!*

*Professional applications programmers will find this versatile utility and it's associated functions invaluable.*

*Comes complete with a substantial 78 page manual and demo programs.*

**Only $49.95!** (State Turbo C or Quick C version.)

**ONLY $49.95 EACH**

## WINDOWS

*Add super-fast text screen handling to your applications with the WINDOWS library from Zortech Inc.*

*Give your applications the professional look – with instant zooming and exploding windows. Incorporate drop-down menus and Lotus style menus with our easy to use functions.*

*Automatically handles memory saving and buffering of window text. Use any number of overlapping windows in your applications. Write to any window, read from any window, close any window, pull any window to the top.*

*Over 55 functions together with a big 85 page manual and remember, you get the full source code.*

**Only $49.95!** (State Turbo C or Quick C version.)

## NEW! C VIDEO

- *Now learn C the easy way!*
- *Get the 'Complete C Video Course' from Zortech Inc. together with our big 220 page workbook.*
- *Ten 1 hour tapes – 36 lessons!*
- *Easy to follow course, you get an excellent introduction to the C language.*
- *Takes you step-by-step up to the intermediate and advanced levels.*
- *Teach yourself at home or the office – at your own speed.*

**only $295.00!**

**Yes! Rush me these items!**

- ☐ HOTKEY
- ☐ COMMS
- ☐ PRO-SCREEN
- ☐ WINDOWS
- ☐ GAMES
- ☐ SUPERTEXT
- ☐ C VIDEO

**FREE SHIPPING – VISA/MC/COD/CHECK**

**Name** .....

**Address** .....

..... **Phone** .....

**VISA or MC #** ..... **Exp. Date** .....

**ZORTECH Inc. 361 Massachusetts Ave, Arlington, MA 02174**
**Orders & Enquiries Tel: (617) 646 6703**

**ORDER HOTLINE (617) 646 6703**

ZORTECH
BOSTON LONDON FRANKFURT GENEVA

**Remember,**



**Smart Buying Decisions**




**Start with DDJ**

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT #200, DALTON, MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER

Reader Service Dept.
P.O. Box 507
Dalton, Mass. 01227

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT #200, DALTON, MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER

Reader Service Dept.
P.O. Box 507
Dalton, Mass. 01227

Use this card for FREE, FAST information about the products and services listed in this issue. Simply circle the appropriate numbers below.

Name _____
Title _____
Company _____ Phone ( ___ ) _____
Address _____
City/State/Zip _____

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 26 | 51 | 76 | 101 | 126 | 151 | 176 | 201 | 226 | 251 | 276 | 301 | 326 | 351 | 376 |
| 2 | 27 | 52 | 77 | 102 | 127 | 152 | 177 | 202 | 227 | 252 | 277 | 302 | 327 | 352 | 377 |
| 3 | 28 | 53 | 78 | 103 | 128 | 153 | 178 | 203 | 228 | 253 | 278 | 303 | 328 | 353 | 378 |
| 4 | 29 | 54 | 79 | 104 | 129 | 154 | 179 | 204 | 229 | 254 | 279 | 304 | 329 | 354 | 379 |
| 5 | 30 | 55 | 80 | 105 | 130 | 155 | 180 | 205 | 230 | 255 | 280 | 305 | 330 | 355 | 380 |
| 6 | 31 | 56 | 81 | 106 | 131 | 156 | 181 | 206 | 231 | 256 | 281 | 306 | 331 | 356 | 381 |
| 7 | 32 | 57 | 82 | 107 | 132 | 157 | 182 | 207 | 232 | 257 | 282 | 307 | 332 | 357 | 382 |
| 8 | 33 | 58 | 83 | 108 | 133 | 158 | 183 | 208 | 233 | 258 | 283 | 308 | 333 | 358 | 383 |
| 9 | 34 | 59 | 84 | 109 | 134 | 159 | 184 | 209 | 234 | 259 | 284 | 309 | 334 | 359 | 384 |
| 10 | 35 | 60 | 85 | 110 | 135 | 160 | 185 | 210 | 235 | 260 | 285 | 310 | 335 | 360 | 385 |
| 11 | 36 | 61 | 86 | 111 | 136 | 161 | 186 | 211 | 236 | 261 | 286 | 311 | 336 | 361 | 386 |
| 12 | 37 | 62 | 87 | 112 | 137 | 162 | 187 | 212 | 237 | 262 | 287 | 312 | 337 | 362 | 387 |
| 13 | 38 | 63 | 88 | 113 | 138 | 163 | 188 | 213 | 238 | 263 | 288 | 313 | 338 | 363 | 388 |
| 14 | 39 | 64 | 89 | 114 | 139 | 164 | 189 | 214 | 239 | 264 | 289 | 314 | 339 | 364 | 389 |
| 15 | 40 | 65 | 90 | 115 | 140 | 165 | 190 | 215 | 240 | 265 | 290 | 315 | 340 | 365 | 390 |
| 16 | 41 | 66 | 91 | 116 | 141 | 166 | 191 | 216 | 241 | 266 | 291 | 316 | 341 | 366 | 391 |
| 17 | 42 | 67 | 92 | 117 | 142 | 167 | 192 | 217 | 242 | 267 | 292 | 317 | 342 | 367 | 392 |
| 18 | 43 | 68 | 93 | 118 | 143 | 168 | 193 | 218 | 243 | 268 | 293 | 318 | 343 | 368 | 393 |
| 19 | 44 | 69 | 94 | 119 | 144 | 169 | 194 | 219 | 244 | 269 | 294 | 319 | 344 | 369 | 394 |
| 20 | 45 | 70 | 95 | 120 | 145 | 170 | 195 | 220 | 245 | 270 | 295 | 320 | 345 | 370 | 395 |
| 21 | 46 | 71 | 96 | 121 | 146 | 171 | 196 | 221 | 246 | 271 | 296 | 321 | 346 | 371 | 396 |
| 22 | 47 | 72 | 97 | 122 | 147 | 172 | 197 | 222 | 247 | 272 | 297 | 322 | 347 | 372 | 397 |
| 23 | 48 | 73 | 98 | 123 | 148 | 173 | 198 | 223 | 248 | 273 | 298 | 323 | 348 | 373 | 398 |
| 24 | 49 | 74 | 99 | 124 | 149 | 174 | 199 | 224 | 249 | 274 | 299 | 324 | 349 | 374 | 399 |
| 25 | 50 | 75 | 100 | 125 | 150 | 175 | 200 | 225 | 250 | 275 | 300 | 325 | 350 | 375 | 400 |

February '88: Use before May 31, 1988

1. Did you buy this issue on a newsstand? ( ) Yes ( ) No
2. Are you a subscriber? ( ) Yes ( ) No
3. Have you purchased a product as a result of seeing it advertised in *Dr. Dobb's Journal?* ( ) Yes ( ) No

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER

Use this card for FREE, FAST information about the products and services listed in this issue. Simply circle the appropriate numbers below.

Name _____
Title _____
Company _____ Phone ( ___ ) _____
Address _____
City/State/Zip _____

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 26 | 51 | 76 | 101 | 126 | 151 | 176 | 201 | 226 | 251 | 276 | 301 | 326 | 351 | 376 |
| 2 | 27 | 52 | 77 | 102 | 127 | 152 | 177 | 202 | 227 | 252 | 277 | 302 | 327 | 352 | 377 |
| 3 | 28 | 53 | 78 | 103 | 128 | 153 | 178 | 203 | 228 | 253 | 278 | 303 | 328 | 353 | 378 |
| 4 | 29 | 54 | 79 | 104 | 129 | 154 | 179 | 204 | 229 | 254 | 279 | 304 | 329 | 354 | 379 |
| 5 | 30 | 55 | 80 | 105 | 130 | 155 | 180 | 205 | 230 | 255 | 280 | 305 | 330 | 355 | 380 |
| 6 | 31 | 56 | 81 | 106 | 131 | 156 | 181 | 206 | 231 | 256 | 281 | 306 | 331 | 356 | 381 |
| 7 | 32 | 57 | 82 | 107 | 132 | 157 | 182 | 207 | 232 | 257 | 282 | 307 | 332 | 357 | 382 |
| 8 | 33 | 58 | 83 | 108 | 133 | 158 | 183 | 208 | 233 | 258 | 283 | 308 | 333 | 358 | 383 |
| 9 | 34 | 59 | 84 | 109 | 134 | 159 | 184 | 209 | 234 | 259 | 284 | 309 | 334 | 359 | 384 |
| 10 | 35 | 60 | 85 | 110 | 135 | 160 | 185 | 210 | 235 | 260 | 285 | 310 | 335 | 360 | 385 |
| 11 | 36 | 61 | 86 | 111 | 136 | 161 | 186 | 211 | 236 | 261 | 286 | 311 | 336 | 361 | 386 |
| 12 | 37 | 62 | 87 | 112 | 137 | 162 | 187 | 212 | 237 | 262 | 287 | 312 | 337 | 362 | 387 |
| 13 | 38 | 63 | 88 | 113 | 138 | 163 | 188 | 213 | 238 | 263 | 288 | 313 | 338 | 363 | 388 |
| 14 | 39 | 64 | 89 | 114 | 139 | 164 | 189 | 214 | 239 | 264 | 289 | 314 | 339 | 364 | 389 |
| 15 | 40 | 65 | 90 | 115 | 140 | 165 | 190 | 215 | 240 | 265 | 290 | 315 | 340 | 365 | 390 |
| 16 | 41 | 66 | 91 | 116 | 141 | 166 | 191 | 216 | 241 | 266 | 291 | 316 | 341 | 366 | 391 |
| 17 | 42 | 67 | 92 | 117 | 142 | 167 | 192 | 217 | 242 | 267 | 292 | 317 | 342 | 367 | 392 |
| 18 | 43 | 68 | 93 | 118 | 143 | 168 | 193 | 218 | 243 | 268 | 293 | 318 | 343 | 368 | 393 |
| 19 | 44 | 69 | 94 | 119 | 144 | 169 | 194 | 219 | 244 | 269 | 294 | 319 | 344 | 369 | 394 |
| 20 | 45 | 70 | 95 | 120 | 145 | 170 | 195 | 220 | 245 | 270 | 295 | 320 | 345 | 370 | 395 |
| 21 | 46 | 71 | 96 | 121 | 146 | 171 | 196 | 221 | 246 | 271 | 296 | 321 | 346 | 371 | 396 |
| 22 | 47 | 72 | 97 | 122 | 147 | 172 | 197 | 222 | 247 | 272 | 297 | 322 | 347 | 372 | 397 |
| 23 | 48 | 73 | 98 | 123 | 148 | 173 | 198 | 223 | 248 | 273 | 298 | 323 | 348 | 373 | 398 |
| 24 | 49 | 74 | 99 | 124 | 149 | 174 | 199 | 224 | 249 | 274 | 299 | 324 | 349 | 374 | 399 |
| 25 | 50 | 75 | 100 | 125 | 150 | 175 | 200 | 225 | 250 | 275 | 300 | 325 | 350 | 375 | 400 |

February '88: Use before May 31, 1988

1. Did you buy this issue on a newsstand? ( ) Yes ( ) No
2. Are you a subscriber? ( ) Yes ( ) No
3. Have you purchased a product as a result of seeing it advertised in *Dr. Dobb's Journal?* ( ) Yes ( ) No

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER

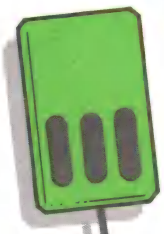
# For Free Info ...

## Start Here

Smart buyers start with *DDJ*'s free information card, a shopping center filled with information about the products and services advertised in this very issue: everything from software and systems to peripherals and professional support services.

And smart buyers can use this free information card to quickly and easily gather a comprehensive file of facts, figures and product specs to sort out competing claims. Using *DDJ*'s free information card can prevent you from making the wrong, costly buying decision.

Be a smart shopper. Complete and mail this postage paid card today!



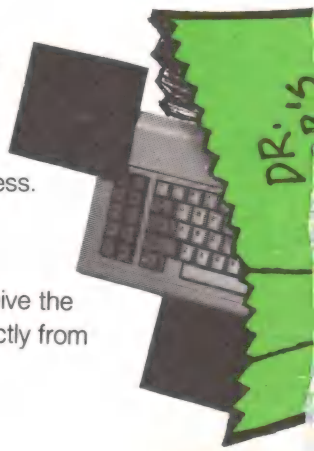
## Take a Reader Service Card with You

## It's Easy as ...

**1.** Circle the appropriate free information numbers, referring to the advertiser index for more information.

**2.** Fill in your name and address.

**3.** Mail today—postage is absolutely free. You'll receive the product information you need directly from the manufacturers, thanks to *DDJ*.



# The Advertiser Index



*The advertiser prefers to be contacted by phone; consult ad.

*(continued from page 126)*

# Microsoft Macro Assembler 5.0

**Target:**
*PC or PS/2 and compatibles*
**Requires:**
*DOS 2.0 or later, hard disk recommended. Minimum 256K memory, 320K recommended*
**Price:**
*$150*
*No licensing restrictions on generated code*
**Vendor:**
*Microsoft 16011 NE 36th Wy, Box 97017, Redmond, WA 98073 (206) 882-8080*

The latest release of Microsoft's venerable MASM brings a new level of productivity to this most tedious of all programming languages. It's loaded with new features, most importantly a bundled copy of the CodeView symbolic debugger.

Microsoft claims that MASM 5.0 assembles 25 percent to 40 percent faster than 4.0. I didn't validate this with any test suites, but it is screamingly fast on an AT: under two seconds for a 150-line program and about five for an 800-liner, which is noticeably faster than earlier versions of the assembler.

With release 5.0, MASM now supports the instruction sets for all PC processors up through the 80386/387. The default mode of the assembler is the 8086 instruction set; pseudo-ops such as .386 and .387 tell the assembler to honor instructions for the indicated targets. If you forget to include the appropriate directive, the assembler scores a severe error for each non-8086 instruction and issues a complaint.

Speaking of pseudo-ops, one potential gotcha for those migrating upward to 5.0 has to do with the encoding of floating point numbers. Before, MASM encoded real numbers in Binary format. The default in 5.0 is the IEEE floating point standard, compatible with 80x87 math

coprocessors. If you still want Microsoft binary format, you have to assert the new .MSFLOAT directive.

Yet another pseudo-op called .MODEL implements memory models, new with MASM 5.0. Its operands are SMALL, MEDIUM, COMPACT, LARGE, and HUGE, which correspond to the models available in high-level Microsoft languages such as C and Pascal, as well as in Turbo C and, to a lesser extent, Turbo Pascal 4.0. This addition simplifies the assembler interface to other languages, since it generates pointers of the correct size for the high-level model being used. Clearly, the emphasis is shifting away from assembler as a primary language to assembler as a language for linked high-performance subroutines, and this addition to MASM recognizes the trend.

To that end, MASM 5.0 furnishes a macro library for mixed-language programming. This file, MIXED.INC, includes macros for passing and receiving arguments, allocating local variables on the stack, segment fixups, exit processing, and other high-level interfacing needs. The macro library is propped up by an excellent 140-page guide covering a range of mixed-language programming issues.

The overall quality of the manuals is much better than any previous Microsoft documentation. In addi-

```
Turbo C » C » Functions » ...
See also:  strncpy()

strcpy()                  Copy One String to Another

#include   <string.h>                  Required for declarations only

char       *strcpy(string1,string2);
char       *string1;                   Destination string
char       *string2;                   Source string


void  addDrive ( char *path )          /* add drive desig to path */
{
char   temp [MAXPATH];                       /* temporary string */
int    p = 0;

  if ( path [1] != ':' ) {
    strcpy ( temp, path );
    path [p++] = disk + 65;             /* convert to A, B, etc. */
    path [p++] = ':';                          /* insert colon */
    if ( temp [0] != '\\' )
      path [p++] = '\\';                       /* and backslash */
    strcpy ( &path [p], temp );             /* restore the rest */
 F1-Help  F5-Zoom  F6-Message  F9-Make  F10-Main menu
```

***Figure 2:*** *MASM 5.0 includes Microsoft's CodeView debugger.*

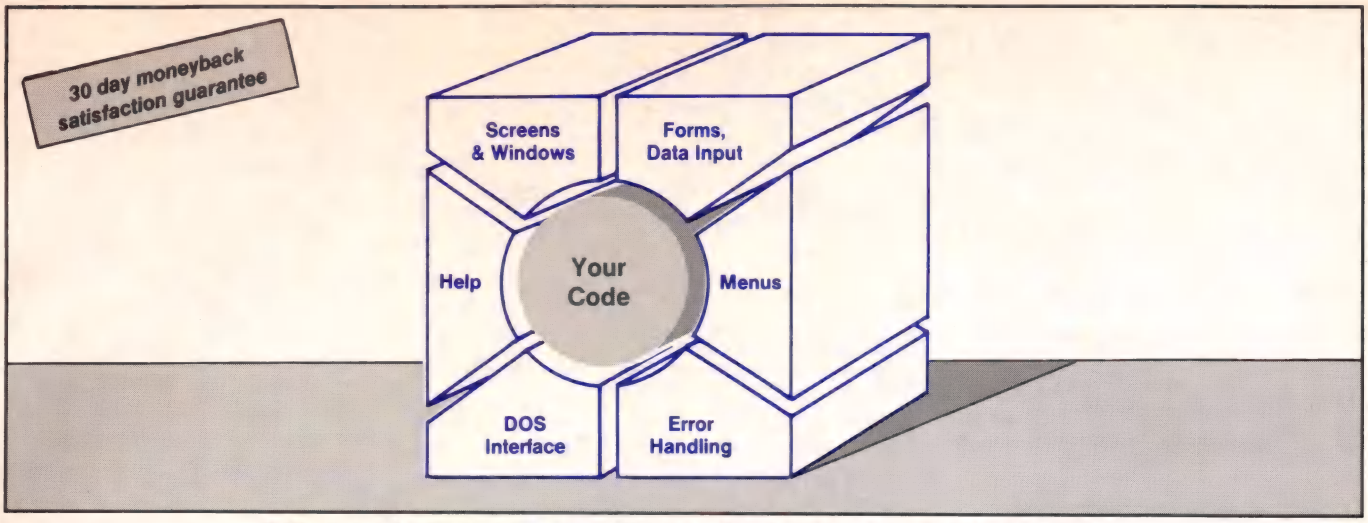
tion to the mixed-language guide, there are two soft-bound books, a 467-page Programmer's Guide and a 401-page volume covering CodeView and other utilities. Rounding out the documentation is a 148-page wire-bound reference divided into five tabbed sections. Intended to be kept at the programmer's elbow, this reference summarizes every instruction, pseudo-op, and command-line switch. There are several brochures and a function-key template for operating CodeView.

If you haven't worked with CodeView before, you're missing out on the finest thing Microsoft has done to date. This is how God intended a symbolic debugger to be. It's a wonderfully visual environment that lets you watch your code execute while keeping an eye on registers and selected memory locations. CodeView runs in a secondary video page, thus not interfering with programs that do graphics or fancy text screens, and you can easily toggle back and forth between the CodeView display and your program's output. Accommodating to all and sundry, CodeView lets you control it via pull-down menus, function keys, typed commands, or a mouse: your choice.

An example of the possibilities with this debugger is a thing called a watchpoint. You can tell CodeView to watch a memory location or even a register and, when it changes to a specified value (or changes at all), CodeView halts and reports the condition. If you've got several watchpoints and breakpoints set, the CodeView display twinkles like Times Square on Friday night, but without the chaos; you are actually watching your program run.

CodeView is not specific to MASM 5.0. It works with all of Microsoft's flagship languages: C, FORTRAN, Pascal, and QuickBASIC. All that's required is to compile and link with certain options, and then you can debug an entire application written in any combination of these languages. Alas, it doesn't work with Turbo languages, but that's hardly surprising.

MASM 5.0 doesn't reduce the notoriously finicky detail of program-




*C-Worthy Interface Library helps you smoothly pull together all aspects of an excellent Human Interface.*

# C Programmers: Wrap an Exciting, Bullet-Proof Interface Around Your Code Quickly.

## *Introducing... C-Worthy® Interface Library*

The only human interface package you need. That's what our customers are telling us. One early adopter, Novell, Inc. uses it exclusively in the development of their NetWare® Utilities, which reach over 500,000 users. You see, C-Worthy Interface Library is the only library available to handle every aspect of your program's human interface, all in one package. Now your programs will have a consistent look and feel. You no longer have to integrate pieces of libraries from different manufacturers.

As important as you know users are, you often don't have the time to heavily invest in writing routine code. And that's OK, because with over 400 tight, ready-to-use functions, C-Worthy Interface Library takes care of the tedium and lets you spend your time doing what you enjoy. Concentrate on the heart of your application — features that make it unique, special. Let C-Worthy Interface Library do your:

- Menus
- Error Handling
- DOS Interface
- Context Sensitive Help
- Screens, Windows
- Forms, Data Input (optional)

You control color, size, border, location, etc. And if there's anything you want to change, you can. Source is available to provide you with the flexibility you need. And you can distribute your applications freely, with no royalties.

C-Worthy Interface Library requires hard disk media with 256K RAM. MSDOS 2.0 + and IBM PC, or compatible, TI Professional, NEC APC III, or VICTOR 9000. C-Worthy is a registered trademark of Custom Design Systems, Inc.

## Tech Specs

- Compilers: Microsoft 3.0+, Quick, Turbo, Lattice. All models.
- 350+ functions written in C, 75+ in Assembler.
- Menus: Fully support pop-up, Lotus style, MS Windows style (pull-down), pull-up.
- Errors: DOS, program, and user.
- DOS Interface: 62 functions. File handling, dir. and drive management, date & time conversion, wildcards, more.
- Help: System and context sensitive.
- Screens: Screen display, color palettes, save, restore, scroll, more.
- Windows: Exploding, tiled, pop-up, overlapping. Direct video access and virtual. Up to 50 active at any time.
- Keyboard Handling: Regular, function, interrupt, background procedures.
- Editing: String and word wrap text.
- Form Interface Library: 118 functions. Over 15 field types, and user definable field types. 3 levels of data validation: type, multiple field ranges, optional validation procedures. Hide, lock, or secure a field. Optimal field movement.
- Foreign Languages: All text messages in separate files for easy translation.
- Compatible with MS Windows.
- OS/2 special overlay when released.
- Machines: Autodetect for MDA, CGA, EGA, VGA, TI, AT&T, Victor.
- No royalties.

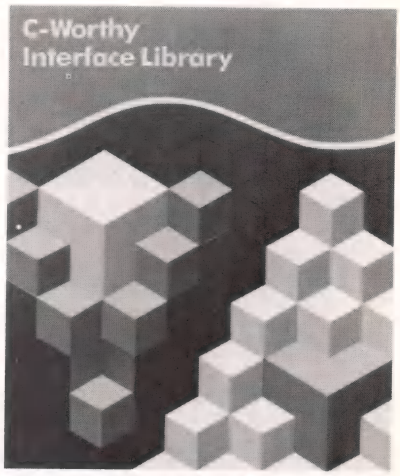
*"I heartily recommend this package."*
*— David A. Schmitt, president, Lattice, Inc.*
*Over 400 developers in 16 countries already use it.*

## Thorough Documentation

Indexed alphabetically and by category, the 700+ page Reference Guide includes for each function: an example, description, calling conventions, return values, and related functions. The 250 page User's Guide gets you going with its tutorial and "Getting Started" sections.

CIRCLE NO. 191 ON READER SERVICE CARD

*"C-Worthy is a comprehensive C library whose time has come. I heartily recommend it as your next purchase." —Computer Language, 8/87*




**C-Worthy Interface Library:**

| | |
|---|---|
| **Object only** | **$ 195** |
| **Form Interface Library add-on** | **$ 100** |
| **Object with Forms** | **$ 295** |
| **Object with Forms & Library Source** | **$ 495** |

*Please specify compiler and version when ordering.*

To Order Call

**(800) 821- 2492**

in MA (617) 337-6963

Solution Systems™

541-D Main Street, Suite 410
South Weymouth, MA 02190

*(continued from page 132)*

ming in assembly language, but it makes for more productive tedium.

*by Kent Porter*

## CheckMate

**Target:**
*PC or PS/2 and compatibles*
**Requires:**
*DOS 2.0 or later, hard disk recommended*
**Price:**
*Annual fees for single-user, department, and site licenses start at $5,750; demo version available.*
**Vendor:**
*Cinnabar Software*
*2704 Rio Grande, Ste. 1, Austin, TX 78705 (512) 477-3212*

CheckMate is a quality assurance tool which runs on a PC or compatible allowing a tester to easily set up automatic scripts for testing communications. CheckMate is intended to test the host computer or communications service (hardware or software) to which the PC is attached.

At the highest level, CheckMate consists of a terminal program. Unfortunately, CheckMate does not emulate any of the popular intelligent terminals such as the VT100 (this feature is scheduled for inclusion in the summer of '88). It does, however, provide a lot more control than is normal with such programs. All communication parameters are accessible. In addition, a high level screen log and a detailed screen trace record all that is happening, both from the keyboard and from the communications ports. Turn your modem off or drop carrier and its instantly recorded on the trace.

CheckMate also provides a capture mode. In this mode, the user goes through a normal communications sequence. CheckMate makes detailed notes on what was entered at the keyboard and what was received at the COM ports, writing this information into a "script" file. During replay, CheckMate makes the keyboard entries and compares the results with what it expects. CheckMate also notes changes in line status such as Carrier Detect or changes in baud rate. The executing sequence can be viewed in progress from the trace screen. Deviations from the expected are noted in the log file. Both the log and the trace can be recorded on disk for later perusal or as an testing audit trail.

CheckMate is designed to allow the user to set up scripts for testing every feature of his product. With such scripts, the user can build a test suite for acceptance testing of the program or service. This suite can be built upon as each new generation is developed. In this way, the tester is assured that no existing features have been "broken" by new additions. These scripts can also be used as a routine status check or to "attack" the system, running the same sequence over and over until a failure occurs.

Unlike most other communications software, however, CheckMate's script files are recorded in Microsoft 4.0 compatible C source code (an example script, created by CheckMate but with comments provided by Cinnabar, appears in Listing One). To play a script back, the user must exit CheckMate, compile and link the C program into an executable file that can then be played back from within the program. CheckMate is intended to handle large, carefully designed scripts which the user might run hundreds of times. The time to compile might be a nuisance during the actual development if it takes multiple iterations to get the script right.

For this, the user gets some pretty significant advantages. Compiled scripts run faster with better timing control than an interpretive script can. Roughly half of the user manual is devoted to a description of the system calls available to the CheckMate programmer. In addition, CheckMate's scripts can be edited by a C programmer to add any extra features desired. The complete C language is at the programmer's disposal.

While not for everyone, CheckMate should be an interesting prod-

# Why We're Betting a Million Lines of Code on the SAS/C™ Compiler.

At SAS Institute Inc., we've invested more than 10 years of research—and over a million lines of code—in the SAS® System, the world's leading data analysis software. So you can bet we left nothing to chance when we chose the C language for the next generation of our software.

We selected C for the portability it would bring to the SAS System, but weren't about to risk our code on just **any** mainframe C compiler. So we tried them all. When none could meet our exacting requirements, we created our own: the SAS/C compiler.

## We Developed It.

## Support It. Use It.

The SAS/C compiler set new standards for efficiency and technical quality, with:

- A source-level debugger that includes structure display, ABEND recovery, and debugger I/O exits for debugging specialized applications
- Reentrant object code
- Highly optimized generated code
- Use of standard IBM linkage conventions, with support for 31-bit addressing
- A CMS Rexx/TSO CLIST interface
- Support for signal handling including program checks and terminal interrupts, and non-standard signals such as timer interrupts and stack overflow
- Many built-in functions including string handling
- In-line assembler.

And when we combined these features with outstanding technical support and frequent updates—both provided free—software developers everywhere took notice. The SAS/C compiler is now the market leader, installed in hundreds of commercial firms and academic institutions.

## Test It. Compare It.

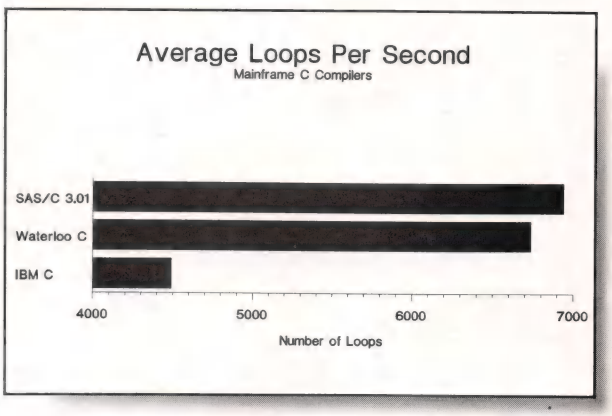
## FREE for 30 Days.

We're betting you've set the same high standards. That's why we'd like to send you the SAS/C compiler, under OS or CMS, for a free 30-day evaluation. We'll also send you a free copy of a leading benchmark program. Compare our compiler with any other. Odds are, you'll choose the SAS/C compiler.

Just mail the coupon below. Or call your Software Sales Representative at (919) 467-8000.

SAS Institute Inc.
SAS Circle ☐ Box 8000
Cary, NC 27512-8000
Phone (919) 467-8000
Fax (919) 469-3737

*Using a C version of the Dhrystone benchmark, the latest SAS/C compiler release produces the fastest code among the top 3 mainframe compilers. It even tops our own previous release by 35%.*




I'd like to put the SAS/C™ compiler to the test with a free 30-day trial, and my free copy of the Dhrystone benchmark program. Give me the details.

*Please complete, or attach your business card.*

Name _____ Title _____

Company _____

Address _____

City _____ State _____ ZIP _____

Telephone _____

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA, 27512-8000

*(continued from page 143)*

uct to those companies which provide communications services or which make heavy internal use of serial communications. CheckMate can also be used to test mainframe software by simulating multiple users logged into serial terminals. CheckMate comes on two floppies within a 8.5-by 11-inch 3-ring binder.

The sitngle CPU lvicense fee for CheckMate is $5,750 per year. A demo version can be ordered for $49.95. CheckMate requires 320k and DOS 3.1 or better to run.

*by Randy Davis*

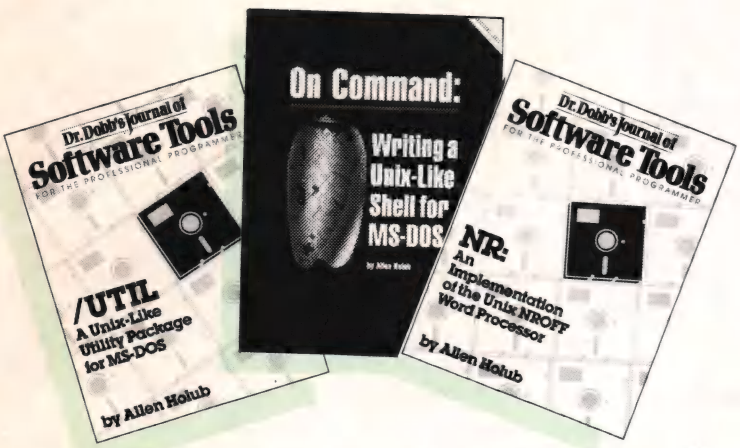
**DDJ**

```
Check*Mate header      */


main(argc, argv)
int argc;


char *argv[];

{
  int  length;                          /* Declare program vaiables.      */
  int  prevwait;
  /* Set online requirements for the Hayes modem.                          */

    PM_set(COM1, _SXDCE, SXCTS);
 /* Set error action parameter to exit the script if any errors occur. */

 PM_set(COM1, _ERRACTION, EAEXIT);
 SX_assert(COM1);                      /* Raise dataset signals and...       */
 MX_online(COM1);                      /* match for online on COM1.          */
 TX_str(COM1, "AT", YES);              /* Send Hayes attention command...    */
 MX_blk(COM1, "OK", 2, 10, &length);/* and match for "OK" returned.       */
 CM_delay(0.5);                        /* Wait for Hayes modem to catch up. */
 TX_str(COM1, "AT DT", NO);            /* Send Hayes command to dial tone    */


 /* Send the phone number for the modem to dial, which is accessed by     */
 /* the script with the string 'argv[1]'.                                 */


 TX_str(COM1, argv[1], YES);

 /* Match "CONNECT" back from modem signaling that host answered.         */

     PM_read(COM1, _WAIT, &prevwait);

     PM_set(COM1, _WAIT, 60);

     MX_blk(COM1, "CONNECT", 7, 40, &length);

     PM_set(COM1, _WAIT, prevwait);

     exit(OK);                                /* Exit script with success      */

}
```

***Example 1:*** *C source code generated by CheckMate.*

# Bring the Conveniences of UNIX To YOUR MS-DOS Machine

## *Full Source Code on Disk!*




### On Command: Writing a Unix-Like Shell for MS-DOS
by Allen Holub

This book and ready-to-use program demonstrate how to write a Unix-like shell for MS-DOS, with techniques applicable to most other programming languages as well. The book and disk include a detailed description and working version of the Shell, complete C source code, a thorough discussion of low-level MS-DOS interfacing, and significant examples of C programming at the system level.

Supported features include: read, aliases, history, redirection and pipes, Unix-like command syntax, MS-DOS-compatible prompt support, C-Shell-based shell scripts, and a Shell variable that expands to the contents of a file so a program can produce text that is used by Shell scripts. The Unix-like control flow includes: if/then/else; while; foreach; switch/case; break; continue.

The ready-to-use program and all C source code are included on disk. For IBM PC and direct compatibles.

**Book & Disk (MS-DOS)** **Item #29-1** **$39.95**

### \Util
by Allen Holub

When used with the Shell, this collection of utility programs and subroutines provides you with a fully functional subset of the Unix environment. Many of the utilities may also be used independently. You'll find executable versions of cat; cp; date; du; echo; grep; ls; mkdir; mv; p; pause; printevn; rm; rmdir; sub; and chmod.

The \Util package includes complete source code on disk and all programs (and most of the utility subroutines) are fully documented in a Unix-style manual. For IBM PC and direct compatibles.

**Manual & Disk (MS-DOS)** **Item #12-7** **$29.95**

### NR: An Implementation of the Unix NROFF Word Processor
by Allen Holub

**NR** is a text formatter that is written in C and compatible with UNIX's NROFF. Complete source code is included in the **NR** package so that it can be easily customized to fit your needs. **NR** also includes an implementation of the -ms (manuscript) macro package and an in-depth description of how -ms works. **NR** does hyphenation and simple proportional spacing. It supports automatic table of contents and index generation, automatic footnotes and endnotes, italics, boldface, overstriking, underlining, and left and right margin adjustment. **NR** also contains:

- extensive macro and string capability
- number registers in various formats, including Roman and Arabic numerals, both spelled out and in outline form
- diversions and diversion traps (macros that are triggered automatically)
- input and output line traps

**NR** is easily configurable for most printers. Both the ready-to-use program and full source code are included.

For PC compatibles.

**Manual & Disk (MS-DOS)** **Item #33-X** **$29.95**

## Save 15%

Receive **On Command, Util** and **NR** together for only $85.95! You get all the convenience of Unix-like features and save 15%.

**Unix-like Features Package** **Item #167** **$85.95**

## To Order:

Return this order form with your payment to:
M&T Books, 501 Galveston Dr., Redwood City, CA 94063
Or, **CALL TOLL-FREE 800-533-4372** Mon-Fri 8AM-5PM
(In CA call 800-356-2002)

### ORDER FORM

Name _____

Address _____

City _____ State _____ Zip _____

***Yes!*** I want to SAVE 15%! Please send me the

**Unix-Like Features Package** for only $85.95 _____

Send me **On Command** book & disk $39.95 _____

**Util** manual & disk $29.95 _____

**NR** manual & disk $29.95 _____

Subtotal _____

CA residents add sales tax ____% _____

Add $2.25 per item for shipping _____

TOTAL _____

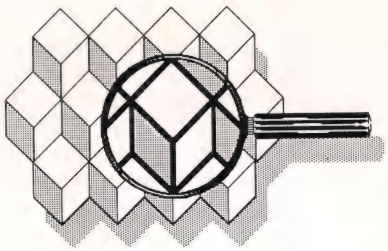
☐ Check enclosed. Make payable to M&T Publishing.

Charge my ☐ VISA ☐ M/C ☐ Am. Ex.

Card No. _____ Exp. _____

3136C

# OF INTEREST



## Debuggers

**Gimpel Software** has announced Turbo C-terp, a debugging C interpreter that is fully compatible with Turbo C. Breakpoints can be temporary or sticky; can be specified via line number, function name, or cursor; and may be conditional. Users can step to the next statement, step over functions, and leave the current function; display the value of any expression; and execute any C expression. C-terp supports full K&R (with ANSI enhancements) and multiple modules and includes a built-in multifile editor, automatic make, virtual memory option, and shared symbols. The product is priced at $139. Reader Service No. 16.
Gimpel Software
3207 Hogarth Ln.
Collegeville, PA 19426
(215) 584-4261

dBFind is a dBASE II, dBASE III, and dBASE III Plus debugger from **The Software Development Factory**. The debugger identifies missing keywords, arguments, and operators; locates unbalanced control structures; builds a complete index of variables by source module and by line number and includes type of usage for each occurrence; and provides command-specific, detailed error messages. The price for dBFind is $99. Reader Service No. 17.
The Software Development Factory
400 E. Pratt St., Ste. 800
Baltimore, MD 21202
(301) 666-8129

**Arium Corp.** has introduced a high-level C language symbolic debugging feature for its Echo microprocessor development system C compiler option. This feature allows design engineers or programmers to debug emulated code in the source language, in assembly language, or both. The C compiler option with C language debug feature is available for $495 for the 8-bit unit and $895 for the 16-bit unit.

Echo is a microprocessor development system that features a 20-Mbyte hard disk, a 1.2-Mbyte flexible disk, 1 Mbyte of main memory, a full-screen editor with dedicated keypad, assemblers, linker and loader software, and complete emulation software. Echo is priced at $8,960 for 8-bit processors and at $12,980 for 16-bit processors. Reader Service No. 18.
Arium Corp.
1931 Wright Circ.
Anaheim, CA 92806
(714) 978-9531

## Tools and Utilities

BPTPLUS in C is a B+Tree data manager available from **Sterling Castle** that allows both direct and sequential access to data. The program features multiple nonunique keys, virtual-index capabilities, partial key searches, and the ability to include 40 indices per index file. With the program, users can convert BASIC programs written with B+Tree to C programs and have access to original BASIC index and data files without having to reenter data. BPTPLUS is compatible with Borland's Turbo C; Microsoft C, Versions 3.0 and 4.0; and Lattice C, Version 3.0. Reader Service No. 19.
Sterling Castle
702 Washington St., Ste. 174
Marina del Rey, CA 90292
(800) 722-7853
In Calif. (800) 323-6406
(213) 306-3020

The Cito 286 programming environment is now available from **Fillmore Systems**. Cito consists of an interactive command-line interpreter and macro language and a linking feature that allows object modules to be dynamically linked into the currently running Cito environment. Users can write, edit, compile, and link C procedures from within the Cito environment. The macro interpreter generates executable machine code for macro definitions and entry points to dynamically linked procedures. Cito supports conditional testing, high-level control structures, file I/O, multitasking, variables, and arithmetic. Version 1.0 of Cito runs on IBM PC ATs and compatibles using Xenix, Version 2.0 (System V). The price is $229. Reader Service No. 20.
Fillmore Systems
7200 York Ave. S., Ste. 301
Edina, MN 55435
(612) 831-6984

The C-scape On-Line Reference has been jointly released by **Oakland Group** and **Peter Norton Computing**. The software is designed to work in conjunction with the Instant Access Program of the recently released Norton Guides and Oakland's C-scape Interface Management System. The Norton Guides provides instant on-line reference data for four languages: C, assembly language, BASIC, and Pascal. C-scape is a professional development tool for controlling the user interface of C programs. The C-scape On-Line Reference Guide is available from Oakland Group for $50. The program requires the Norton Guides' Instant Access Program, which can be purchased from Oakland Group or Peter Norton Computing for $50. Reader Service No. 21.
Oakland Group Inc.
675 Massachusetts Ave.
Cambridge, MA 02139-3309
(800) 233-3733
(617) 471-7311

C:Lines/C:Tree, from **SoftRex**, is a program that reorganizes and prints C source code and provides a structural analysis of the code. Features included are outlines of all logical blocks that are added to the listings, detailed cross-references, an automatic source code reformatter, and a hierarchy tree that shows the relationships between the various subroutines in code. C:Lines/C:Tree can

also generate a listing of unused global symbols, preview listings on a monitor, analyze interactions between overlays, add titles and subtitles to listings, generate an index of titles and subtitles, and be configured for any printer. The program runs on IBM PCs and compatibles and sells for $59.95. Reader Service No. 22.
SoftRex
4807 Bethseda Ave., #287
Bethseda, MD 20814
(301) 881-8274

A new coding tool from **O'Neill Software** called the Text Collector lets programmers find, collect, examine, and analyze scattered blocks of source code. The package searches an entire disk or specified groups of files for any combination of terms and automatically saves all context blocks meeting the search criteria. Context blocks can be lines of code, strings, comments, records, or other user-defined blocks of material. File names can be appended or prepended to blocks, and the blocks can be sorted. The program permits complex searches using Boolean operators, nested parentheses, and 14 different wildcards. The Text Collector runs on IBM PCs and compatibles with 128K and MS-DOS/PC-DOS 2.0 or later. The price is $69. Reader Service No. 23.
O'Neill Software
P.O. Box 26111
San Francisco, CA 94126
(415) 398-2255

The Window Management System (WMS) for Turbo C is a high-level programming tool that allows efficient development of easy-to-use interfaces based on windows, menus, and data prompts. **B&C Microsystems** is now shipping this program, which features a window editor, a window librarian, a C interface, and DOS utilities. WMS sells for $69.95. A demo disk is available for $5 and can be credited toward future purchases of WMS packages. Reader Service No. 24.
B&C Microsystems
355 W. Olive Ave.
Sunnyvale, CA 94086
(408) 730-5511

**Serengeti Software** has announced Show Me!, a memory-resident, file-viewing utility for the IBM PS/2, IBM PC, and compatibles. The program allows users to list directories and view up to four files instantly; use pop-up help screens for virtually any program; look at WordStar document files while editing and pass text from a window; view multiple source or listing files in ASCII or hexadecimal while using DEBUG, SYMDEB, or other utilities; and add help screens to programs using built-in program interfaces to BASIC, Pascal, dBASE, and C. Show Me! sells for $39. Reader Service No. 25.
Serengeti Software
P.O. Box 27254
Austin, TX 78755
(512) 345-2211

## Languages

**Addison-Wesley** has released *The AWK Programming Language*, the

first book on awk, a programmable text-manipulation language for writing short programs to perform data-manipulation tasks. The language was originally developed by Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger in 1977. A new version of the language, which is available for Unix and MS-DOS, was developed in 1985. Some of the topics covered in the book include retrieving, transforming, reducing, and validating data; managing small personal databases; text processing; little languages; and experimenting with algorithms. The book sells for $21.95. Reader Service No. 26.
Addison-Wesley
Reading, MA 01867
(617) 944-3700

**MHT Software** is now shipping three new products: the cENGLISH Menu Development System; cENGLISH, Version 3.0; and cENGLISH for Microsoft C. The cENGLISH developer employs English-like commands that are translated into C language source code. This C code uses the industry-standard C-ISAM access manager and in Informix-SQL database management system to organize and retrieve information. The cENGLISH Menu Development System is an interactive utility for building many different styles of menus. cENGLISH, Version 3.0, adds video and printer control, text file manipulation, and utilities for importing and exporting ASCII data. Programmers without C experience can use cENGLISH for Microsoft C and benefit from the advantages of the C language. Prices vary according to machine and operating system. Reader Service No. 27.
MHT Software
2923 Saturn St., Ste. A
Brea, CA 92621
(714) 528-1602

**Software Development Systems** has announced its UniWare Z80 C compiler, which was designed to help engineers make the transition from writing in assembly language to writing in C. A special optimizing feature of the UniWare Z80 C compiler is its ability to use the Z80's alternate register set for storing up to three 16-bit user-declared variables. The compiler comes complete with a macro relocating assembler, a linker, an object module librarian, and utilities to help programmers debug their code and download it into emulators and EPROM programmers. The UniWare Z80 C compiler running on the IBM PC and compatibles using MS-DOS or Xenix is priced at $995. The compiler running under Unix on a variety of machines is priced at $2,995. Reader Service No. 28.
Software Development Systems Inc.
3110 Woodcreek Dr.
Downers Grove, IL 60515
(312) 971-8170

**DDJ**

# Upgrade your technology

**The software technology available to programmers of IBM-compatible personal computers is truly amazing. And newer, more powerful development packages appear all the time. But until now, finding out about these important products has been a difficult and time consuming task.**

**FREE Buyer's Guide.** The New Programmer's Connection Buyers Guide contains individual descriptions of over 500 titles of programmer's development software by over 150 manufacturers. Each description covers major product features as well as any software or hardware requirements and version numbers. In the box on the right are some examples of the types of descriptions you'll find in our Buyer's Guide.

**No Hidden Charges.** The low discount prices in our Buyer's Guide are all you pay. We don't charge extra for domestic UPS Ground shipping, credit cards, COD orders, purchase orders, sales tax (except Ohio) or special handling (except for non-Canadian international orders).

**Guarantees.** We offer FREE 30-day no-risk return guarantees and 30-day evaluation periods on most of our products.

**Latest Versions.** The products we carry are the latest versions and come with the same manufacturer's technical support as if buying direct.

**Large Inventory.** We have one of the largest inventories of programmer's development products in the industry. Most orders are shipped within 24 hours.

**Noncommissioned Staff.** Our courteous salespeople are always ready to help you. And if you aren't sure about your needs, our knowledgeable technical people can give you sound, objective advice.

**Experience.** We've specialized in development software for IBM-compatible personal computers since 1984 and are experienced at providing a full range of quality products and customer services.

**How to Get Your Copy.** There are three ways for you to receive your FREE copy of the Programmer's Connection Buyer's Guide: 1) Use the reader service card provided by this journal; 2) Mail us a card or letter with your name and address; or 3) Call one of our convenient toll free telephone numbers.

**If you haven't yet received your Programmer's Connection Buyer's Guide, act now. Upgrading your programming technology could be one of the wisest and most profitable decisions you'll ever make.**

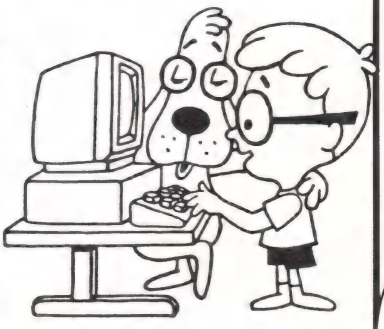
## Copia International

### Peabody

**Pop-up Resident Reference Utility**

**List $100 Reg $89**
**Special Introductory Price $79**
through 02/28/88

Peabody is a fast and flexible on-line reference utility with databases for Microsoft C, Turbo Pascal and MS-DOS. Peabody spares the manual and spoils the programmer by providing instant, accurate and complete information in pop-up frames at the touch of a key. Whether you need a simple reminder or a full description and useful examples, you'll find it with Peabody. Each database has been designed and assembled by seasoned programmers; their applied experience is obvious in the concise and practical information that unfolds at your request in overlapping frames. Peabody gives you two ways of finding that information; select general topics from a structured subject menu or use Peabody's unique Hyper-Key to get instant help for the keyword closest to the cursor. Other special features that make Peabody an outstanding value are its ability to reference multiple databases with a single keystroke; its Sticky Frame feature, which lets you paste information on screen for reference when you return to your editor; and display functions which list on screen the contents of disk files and computer memory in text or binary formats (ASCII or EBCDIC). Peabody can run as a standalone program, in tandem with your text editor, or as a memory-resident utility. The Peabody disk includes a setup utility for customizing Peabody and a compiler for integrating your own subject categories, keywords and reference material directly into a Peabody database.



Requires as little as 85K memory. **Comes with a full unconditional 60-day money-back guarantee.**

## CALL TOLL FREE

**USA:** ..... **800-336-1166**
**Canada:** ..... **800-225-1166**
**Ohio & Alaska (Collect):** .. **216-494-3781**

**International:** **216-494-3781**

**Telex:** ..... **9102406879**
**Easylink:** ..... **62806530**

**Programmer's Connection**
**7249 Whipple Avenue NW**
**North Canton, OH 44720**

**Please turn the page for our latest price list and ordering information.** ➡

PROGRAMMER'S CONNECTION
QUALITY INTEGRITY
Service
SUPPORT PRICE

# The Free Programmer's Connection Buyer's Guide.

CIRCLE NO. 198 ON READER SERVICE CARD

## ai - expert systems

| | List | Ours |
|---|---|---|
| **1st-CLASS** *by Programs in Motion* | 495 | 399 |
| **EXSYS** *Development Software by EXSYS* | 395 | 289 |
| **LEVEL5** *by Information Builders* | 685 | 569 |
| **Logic-Line Series** *All varieties by Thunderstone* | CALL | CALL |
| **RuleMaster 2** *by Radian Corporation* *New* | CALL | CALL |
| **VP-EXPERT** *by Paperback Software* | 125 | 79 |

## ai - lisp language

| | | |
|---|---|---|
| **muLISP-87 Interpreter** *by Soft Warehouse* | 300 | 199 |
| **muLISP-87 Interpreter & Compiler** | 400 | 259 |
| **Q'Nial** *Various by NIAL Systems* | CALL | CALL |
| **Star Sapphire LISP** *Compiler by Sapiens* | 495 | 429 |

## ai - prolog language

| | | |
|---|---|---|
| **Arity Combination Package** | 1095 | 979 |
| **Expert System Development Pkg** | 295 | 229 |
| **File Interchange Toolkit** | 50 | 44 |
| **PROLOG Compiler & Interpreter** | 650 | 569 |
| **Screen Design Toolkit** | 50 | 44 |
| **SQL Development Package** | 295 | 229 |
| **Arity PROLOG Interpreter** | 295 | 229 |
| **Arity Standard Prolog** | 95 | 77 |
| **LPA microPROLOG** *All Varieties* | CALL | CALL |
| **MPROLOG P550** *w/Primer by LOGICWARE* | 220 | 159 |
| **Turbo PROLOG** *by Borland Intl* | 100 | 64 |
| **Turbo PROLOG Toolbox** *by Borland Intl* | 100 | 64 |

## ai - smalltalk language

| | | |
|---|---|---|
| **Smalltalk/V** | 100 | 84 |
| **EGA/VGA Color Option** | 50 | 45 |
| **Goodies Diskette #1** | 50 | 45 |
| **Smalltalk/Comm** | 50 | 45 |

## ai - texas instruments

| | | |
|---|---|---|
| **Arborist** *Decision Tree Software* | 595 | 519 |
| **PC Scheme** *Lisp* | 95 | 77 |
| **Personal Consultant Easy** | 495 | 435 |
| **Personal Consultant Images** | 495 | 435 |
| **Personal Consultant Online** | 995 | 869 |
| **Personal Consultant Plus** | 2950 | 2589 |
| **Personal Consultant Runtime** | 95 | 84 |

## ada language

| | | |
|---|---|---|
| **AdaVantage** *GSA-validated by Meridian Software* | 795 | 735 |
| **AdaVantage Utility Packages** | 50 | 47 |
| **DOS Environment Package** | 50 | 47 |
| **Ada** *GSA-validated w/maintenance by alsys* *New* | 3355 | 3119 |
| **AdaQUERY** *New* | 200 | 185 |
| **Ada Developer's Toolset** *Volumes 1 & 2* *New* | 995 | 919 |

## apl language

| | | |
|---|---|---|
| **APL*PLUS** *by STSC, Specify PC or PS/2* | 695 | 495 |
| **APL*PLUS PC Spreadsheet Mgr** *by STSC* | 195 | 139 |
| **APL Tools** *by STSC* | 295 | 199 |
| **ATLAS*GRAPHICS** *by STSC* | 450 | 329 |
| **Financial/Statistical Library** *by STSC* | 275 | 189 |
| **Pocket APL** *by STSC* | 95 | 69 |
| **STATGRAPHICS** *by STSC* | 895 | 649 |

## assembly language

| | | |
|---|---|---|
| **386/ASM/386 LINK** *Cross Asm by Phar Lap* | 495 | 389 |
| **ASMLIB** *Function Library by BCSoft* | 149 | 125 |
| **asmTREE** *B-Tree Dev System by BCSoft* | 395 | 329 |
| **Cross Assemblers** *Various by 2500 AD* | CALL | CALL |
| **DMS Resident-ASM** *by American Software Intl* *New* | 150 | 139 |
| **Microsoft Macro Assembler** | 150 | CALL |
| **OPTASM** *by SLR Systems* *New* | 195 | CALL |
| **risC** *by IMSI* *New* | 80 | 69 |
| **Turbo Debugger** *by Speedware* | 89 | 79 |
| **Turbo Editasm** *by Speedware* | 99 | 84 |
| **Visible Computer: 8088** *by Software Masters* | 80 | 64 |

## basic language

| | | |
|---|---|---|
| **ApBasic** *by CompTech* *New* | 99 | 79 |
| **db/Lib** *for QuickBASIC by AJS Publishing* | 139 | 119 |
| **Finally** *by Komputerwerk* | 99 | 85 |
| **MACH 2** *by MicroHelp* | CALL | CALL |
| **Microsoft QuickBASIC** | 99 | CALL |
| **QBase** *Relational Database by Crescent* | 99 | 89 |
| **Quick-TOOLS** *by BCSoft* | 130 | 109 |
| **QuickPak** *by Crescent Software* | 69 | 59 |
| **Scientific Subroutine Library** *by Wiley* | 125 | 99 |
| **Screen Sculptor** *by Software Bottling* | 125 | 91 |
| **Stay-Res** *by MicroHelp* | 79 | 53 |
| **True BASIC** | 100 | 69 |
| **True BASIC** *w/Run-time* | 150 | 99 |
| **True BASIC 3D Graphics** | 50 | 41 |
| **True BASIC Developer's Toolkit** | 50 | 41 |
| **Turbo BASIC Compiler** *by Borland Intl* | 100 | 64 |

## blaise products

| | | |
|---|---|---|
| **ASYNCH MANAGER** *Specify C or Pascal* | 175 | 135 |
| **C TOOLS PLUS/5.0** | 129 | 99 |
| **KeyPlayer** *Super Batch Program* | 50 | 45 |
| **LIGHT TOOLS** *for Datalight C* | 100 | 65 |
| **PASCAL TOOLS/TOOLS 2** | 175 | 135 |
| **RUNOFF** *Text Formatter* | 50 | 45 |
| **Turbo ASYNCH PLUS/4.0** | 129 | 99 |
| **Turbo C TOOLS** | 129 | 99 |
| **Turbo POWER TOOLS PLUS/4.0** | 129 | 99 |
| **VIEW MANAGER** *Specify C or Pascal* | 275 | 199 |

## borland products

| | | |
|---|---|---|
| **EUREKA** *Equation Solver* | 167 | 105 |
| **Quattro: The Professional Spreadsheet** *New* | 195 | 125 |
| **Reflex: The Analyst** | 150 | 99 |
| **Sidekick** | 85 | 57 |
| **Superkey** | 100 | 64 |
| **Turbo Basic Compiler** | 100 | 64 |
| **Turbo Basic Database Toolbox** | 100 | 64 |
| **Turbo Basic Editor Toolbox** | 100 | 64 |
| **Turbo Basic Telecom Toolbox** | 100 | 64 |
| **Turbo C Compiler** *(Call for support products)* | 100 | 64 |
| **Turbo Lightning and Word Wizard** | 150 | 94 |
| **Turbo Lightning** | 100 | 64 |
| **Turbo Lightning Word Wizard** | 70 | 47 |
| **Turbo Pascal** | 100 | 64 |
| **Turbo Pascal Database Toolbox** | 100 | 64 |
| **Turbo Pascal Developer's Toolkit** | 395 | 289 |
| **Turbo Pascal Editor Toolbox** | 100 | 64 |
| **Turbo Pascal Gameworks Toolbox** | 100 | 64 |
| **Turbo Pascal Graphix Toolbox** | 100 | 64 |
| **Turbo Pascal Numerical Methods Toolbox** | 100 | 64 |
| **Turbo Pascal Tutor** | 70 | 41 |
| **Turbo Prolog Compiler** | 100 | 64 |
| **Turbo Prolog Toolbox** | 100 | 64 |

## c language

| | | |
|---|---|---|
| **C-terp** *by Gimpel, Specify compiler* | 298 | 219 |
| **C Trainer** *with Book by Catalytix* | 123 | 87 |
| **DC88** *by C Ware* *New* | 99 | 89 |
| **DC88** *with Large Case Option by C Ware* *New* | 178 | 159 |
| **Eco-C88 Modeling Compiler** *by Ecosoft* | 100 | 79 |
| **Instant C** *by Rational Systems* | 495 | 369 |
| **Lattice C Compiler** *vers. 3.2 from Lattice* | 500 | 265 |
| **Mark Williams Let's C** *with FREE csd* | 75 | 54 |
| **Microsoft C Compiler 5.0 w/CodeView** | 450 | CALL |
| **Microsoft QuickC Compiler** | 99 | CALL |
| **Optimum-C** *by Datalight* | 139 | 95 |
| **Turbo C Compiler** *by Borland* | 100 | 64 |
| **Turbo C-terp** *for Turbo C by Gimpel* *New* | 139 | 119 |
| **Uniware 68000/10/20 Cross Compiler** *by SDS* | 995 | 829 |

## c utilities

| | | |
|---|---|---|
| **Blackstar C Library** *by Sterling Castle* | 125 | 98 |
| **C++** *by Guidelines* *New Version* | 295 | 259 |
| **c-tree & r-tree Combo** *by FairCom* | 650 | 519 |
| **c-tree** *ISAM File Manager* | 395 | 315 |
| **r-tree** *Report Generator* | 295 | 239 |
| **Curses Window Dev Pkg** *by Aspen Scientific* | 119 | 105 |
| *with Source Code* | 289 | 249 |
| **dBx** *dBASE to C Translator by Desktop AI* | 350 | 299 |
| *with Source Code* | 550 | 419 |
| **DMS Resident-C** *by American Software Intl* *New* | 150 | 139 |
| **Entelekon Combo Package** *Special Price* | 200 | 99 |
| **Flash-up** *by Software Bottling* | 89 | 78 |
| **GraphiC** *by Scientific Endeavors* *New Version* | 395 | 309 |
| **HALO** *Graphics by Media Cybernetics* | 300 | 205 |
| **HALO Development Pkg** *for Microsoft* | 595 | 389 |
| **The HAMMER** *by OES Systems* | 195 | 129 |
| **Heap I/O** *by System Software* *New* | CALL | CALL |
| **HOOPS** *3-D Graphics by Ithaca Software* *New* | 575 | 459 |
| **LINK & LOCATE** *by Systems & Software* *New* | 350 | 309 |
| **PANEL** *by Roundhill, Specify QuickC or Turbo C* | 129 | 95 |
| **PANEL Plus** *by Roundhill* | 495 | 395 |
| **PC Lint** *by Gimpel Software* | 139 | 99 |
| **RTC PLUS** *Fortran to C by Cobalt Blue* | 450 | 369 |
| **Sapiens V8** *Virtual Memory Manager* | 300 | 265 |
| **Scientific Subroutine Library** *by Wiley* | 175 | 135 |
| **TE Developer's Kit** *by Sub Systems* | 95 | 85 |
| **Vitamin C** *by Creative Programming* | 225 | 149 |
| **VC Screen** *Forms Designer* | 100 | 79 |
| **WKS LIBRARY** *by Tenon Software* *New* | 89 | 79 |
| **Zview** *by Data Mgmt Consultants* | 245 | 119 |

## cobol language

| | | |
|---|---|---|
| **COBOL spll** *by Flexus* | 395 | 329 |
| **Micro Focus COBOL** *See Micro Focus Section* | | |
| **Microsoft COBOL** *See Microsoft Section* | | |
| **Realia COBOL with RealMENU** | 1145 | 899 |
| **Realia COBOL** | 995 | 783 |
| **RealCICS** | 995 | 783 |
| **RM/COBOL** *by Ryan-McFarland* | 950 | CALL |
| **RM/COBOL 85** *by Ryan-McFarland* | 1250 | CALL |
| **RM/USER+5** *by Ryan-McFarland* | 250 | CALL |
| **RM/Screens** | 395 | CALL |
| **SCREENIO** *by Norcom* | 400 | 379 |

## database management

| | | |
|---|---|---|
| **Advanced DBMaster** *by Macon Software* | 510 | 419 |
| **Advanced Revelation** *by COSMOS* *New* | 950 | CALL |
| **Clipper** *by Nantucket* *New* | 695 | 379 |
| **DB-FABS** *by Computer Control Systems* *New* | 295 | CALL |
| **dB2c** *by Software Connection* *New* | 299 | CALL |
| **dBASE III Plus** *by Ashton-Tate* | 695 | 389 |
| **dbSQL** *by WordTech Systems* | CALL | CALL |
| **dBXL** *by WordTech Systems* | 169 | 109 |
| **dFLOW** *by Wallsoft* | 149 | 119 |
| **The Documenter** *by Wallsoft* | 295 | 225 |
| **enable** *by The Software Group* *New* | CALL | CALL |
| **Fox Base Plus** *by Fox Software* *New* | 395 | 249 |
| **Genifer** *by Bytel* | 395 | 275 |
| **MAGIC PC** *by AKER* *Special Price,New* | 199 | 179 |
| **Paradox 1.1** *by Ansa/Borland* | 495 | 359 |
| **Paradox 2.0** *by Ansa/Borland* | 725 | 525 |
| **Paradox Network Pack** *by Ansa/Borland* | 995 | 725 |
| **Q&A** *by Symantec* | 349 | 219 |
| **QUICKCODE PLUS** *by Fox & Geller* | 295 | 169 |
| **QUICKINDEX** *by Fox & Geller* | 149 | 95 |
| **QUICKREPORT** *by Fox & Geller* | 295 | 169 |
| **QuickSilver** *by WordTech Systems* | 599 | 349 |
| **R:Base 5000** *by Microrim* | 495 | CALL |
| **R:Base System V** *by Microrim* | 700 | CALL |
| **\rdb** *by Robinson-Shafer-Wright* | 139 | 119 |
| **SQL Base** *by Gupta Technologies* *New* | 995 | CALL |
| *Multi-User Version* *New* | 1995 | CALL |
| **Tom Rettig's Library** *by Tom Rettig & Assoc* | 100 | 79 |
| **UI Programmer** *by Wallsoft* | 295 | 239 |
| **VP-INFO** *by Paperback Software* | 125 | 79 |
| **VP-PLANNER** *by Paperback Software* *New* | 100 | 63 |
| **VP-PLANNER PLUS** *by Paperback Software* *New* | 125 | 79 |
| **XDB II** *by Software Systems Technology* *New* | 395 | CALL |
| **C Programming Interface** *New* | 295 | CALL |

## debuggers & profilers

| | | |
|---|---|---|
| **386 DEBUG** *Cross Debugger by Phar Lap* | 195 | 129 |
| **Advanced Trace-86** *by Morgan Computing* | 175 | 115 |
| **Codesmith-86** *by Visual Age* | 145 | 98 |
| **DSD87** *by Soft Advances* | 125 | 79 |
| **MiniProbe** *by Atron* | 395 | 369 |
| **Periscope I** *with Board by Periscope* | 345 | 275 |
| **Periscope II** *with NMI Breakout Switch* | 175 | 139 |
| **Periscope II-X** *Software only* | 145 | 105 |
| **Periscope III** *8 MHz version* | 995 | 795 |
| **Periscope III** *10 MHz version* | 1095 | 875 |
| **The PROFILER** *with Source Code by DWB* | 125 | 89 |
| **TURBOsmith** *Source debugger for Turbo Pascal* | 99 | 69 |
| **The WATCHER** *Profiler by Stony Brook* | 60 | 51 |

## disk utilities

| | | |
|---|---|---|
| **Back-It** *by Gazelle Systems* | 130 | 115 |
| **Disk Optimizer** *by Softlogic Systems* | 60 | 55 |
| **Disk Technician** *by Prime Solutions* | 100 | 89 |
| **FASTBACK** *by 5th Generation Systems* | 159 | 115 |
| **FASTBACK PLUS** *by 5th Generation Systems* *New* | 189 | 139 |
| **Take Two Manager** *United Software Security* | 139 | 119 |
| **Vcache** *by Golden Bow Systems* | 50 | 47 |
| **Vopt** *by Golden Bow Systems* | 50 | 47 |
| **Vfeature Deluxe** *by Golden Bow Systems* | 120 | 111 |
| **XenoCopy-PC** *by XenoSoft* | 80 | 69 |

## dos utilities

| | | |
|---|---|---|
| **Advanced Norton Utilities** | 150 | 89 |
| **Command Plus** *by ESP Software* | 80 | 69 |
| **Desqview** *from Quarterdeck* | 130 | 115 |
| **FANSI-CONSOLE** *by Hersey Micro* | 75 | 62 |
| **Mace Utilities** *Paul Mace Software* | 99 | 89 |
| **MicroHelp Utility** *by MicroHelp* | 59 | 49 |
| **Norton Commander** *by Peter Norton* | 75 | 55 |
| **Norton Utilities** *by Peter Norton* | 100 | 59 |
| **Q-DOS II** *by Gazelle Systems* | 70 | 59 |
| **Taskview** *by Sunny Hill Software* | 80 | 55 |
| **The Weiner Shell** *by Gryphon Microproducts* *New* | 199 | CALL |
| **XO-SHELL** *by Wyte Corporation* *New* | 69 | CALL |

## essential products

| | | |
|---|---|---|
| **Breakout Debugger** *by Essential Software* | 125 | 89 |
| **C Utility Library** | 185 | 118 |
| **Essential Communications** | 185 | 118 |
| **Essential Communications** *with Break Out* | 250 | 189 |
| **Essential Graphics** | 250 | 183 |
| **/*residentC*/** | 99 | 85 |
| *with Source Code* | 198 | 148 |
| **ScreenStar** | 99 | 85 |
| *with Library Source Code* | 198 | 154 |

## forth language

| | | |
|---|---|---|
| **FORTH/83 Metacompiler** *Specify Target* | 750 | 599 |
| **MMS Forth** *by Miller Microcomputer Svcs* *New* | 180 | 159 |
| **PC/FORTH** *by Laboratory Microsystems* | 150 | 109 |
| **PC/FORTH+** *by Laboratory Microsystems* | 250 | 188 |
| **Programmer's Package #1** *by LMI* | 250 | 199 |
| **Programmer's Package #2** *by LMI* | 350 | 279 |
| **Programmer's Package #3** *by LMI* | 500 | 399 |
| **UR/FORTH** *Also Available for OS/2 by LMI* | 350 | 258 |
| **UR/FORTH Libraries** | 500 | 395 |

## fortran language

| | | |
|---|---|---|
| **50 MORE: FORTRAN** *by Wiley* | 125 | 95 |
| **ACS Time Series** *by Alpha Computer Service* | 495 | 389 |
| **AUTOMATED PROGRAMMER** *by KGK Automated* | 995 | 899 |
| **Essential Graphics** *by Essential Software* | 250 | 183 |
| **Forlib-Plus** *by Alpha Computer Service* | 70 | 44 |
| **FORTRAN Addenda** *by Impulse Engr* | 165 | 138 |
| **HALO** *Graphics by Media Cybernetics* | 300 | 205 |
| **I/O PRO w/No Limit Library** *by MEF* | 250 | 219 |
| **Microcompatibles Combo Package** | 240 | 215 |
| **Grafmatic** | 135 | 117 |
| **Plotmatic** | 135 | 117 |
| **Microsoft FORTRAN** *w/CodeView* | 450 | CALL |
| **No Limit Library** *by MEF Environmental* | 129 | 109 |
| **Numerical Analyst** *by MAGUS* | 295 | 199 |
| **PANEL** *by Roundhill Computer Systems* | 295 | 199 |
| **RM/FORTRAN** *by Ryan-McFarland* | 595 | 399 |
| **Scientific Subroutine Library** *by Wiley* | 175 | 135 |
| **Statistician** *by Alpha Computer Service* | 295 | 235 |
| **Strings & Things** *by Alpha Computer Service* | 70 | 45 |

## greenleaf products

| | | |
|---|---|---|
| **Greenleaf C Sampler** *specify QuickC or Turbo C* | 95 | 69 |
| **Greenleaf Comm Library** | 185 | 125 |
| **Greenleaf Data Windows Library** | 225 | 155 |
| *with Source Code* | 395 | 249 |
| **Greenleaf Functions** | 185 | 125 |

## help utilities

| | | |
|---|---|---|
| **HELP/Control** *by MDS* | 125 | 99 |
| **On-line Help** *from Opt-Tech* | 149 | 99 |
| **SoftScreen/HELP** *by Dialectic Systems* | 195 | 149 |

## lattice products

| | | |
|---|---|---|
| **Lattice C Compiler** *ver 3.2 from Lattice* | 500 | 265 |
| *with Library Source Code* | 900 | 495 |
| **C Cross Reference Generator** | 50 | 37 |
| **C-Food Smorgasbord** *Function Library* | 150 | 95 |
| *with Source Code* | 300 | 179 |
| **C-Sprite** *Source Level Debugger* | 175 | CALL |
| **Curses** *Screen Manager* | 125 | 85 |
| *with Source Code* | 250 | 169 |
| **dBC III** | 250 | 169 |
| *with Source Code* | 500 | 356 |
| **dBC III Plus** | 750 | 594 |
| *with Source Code* | 1500 | 1184 |
| **LMK** *Make Facility* | 195 | 138 |
| **RPG II Combo** *All four items below* *New Version* | 1400 | 1099 |
| **RPG II Compiler** *No Royalties* *New Version* | 750 | 625 |
| **Screen Design Aid Utility** *for RPG II* | 350 | 309 |
| **SEU** *Source Entry Utility* | 250 | 199 |
| **Sort/Merge** | 250 | 199 |
| **SecretDisk II** *Encryption Utility* | 79 | 59 |
| **SideTalk** *Resident Communications* | 120 | 88 |
| **SSP/PC** *Scientific Subroutine Library* | 350 | 269 |
| **Text Management Utilities** | 120 | 88 |

## metagraphics products

| | | |
|---|---|---|
| **FontWINDOW** | 95 | 79 |
| **LightWINDOW/C** *for Datalight C* | 95 | 79 |
| **MetaWINDOW** *No Royalties* | 195 | 159 |
| **MetaWINDOW/PLUS** | 275 | 229 |
| **TurboWINDOW/C** *for Turbo C* | 95 | 79 |
| **TurboWINDOW/Pascal** *for Turbo Pascal* | 95 | 79 |

## micro focus products

| | | |
|---|---|---|
| **Micro Focus COBOL/2** | 900 | 729 |
| **Micro Focus COBOL/2 Toolset** *New* | CALL | CALL |
| **Micro Focus COBOL/IQ** *Ad hoc Report Writer* | 495 | 395 |
| **Micro Focus COBOL/IQ** *for DOS 3.X Networks* | 995 | 795 |
| **Micro Focus FORMS-2** | 295 | 235 |
| **Micro Focus Level II COBOL** *w/Animator* | 495 | 395 |
| **Level II COBOL** | 349 | 279 |
| **Level II Animator** | 195 | 155 |
| **Micro Focus PC-CICS** | 1495 | 1189 |
| *with* **Micro/SPF** | 1595 | 1269 |
| **Micro Focus Personal COBOL** | 149 | 119 |
| **Micro Focus SOURCEWRITER** | 995 | 795 |
| **Micro Focus VS COBOL/XENIX** | 1495 | 1195 |

## microport products

| | | |
|---|---|---|
| **386 Unlimited License Kit** | 249 | 209 |
| **AT Unlimited License Kit** | 249 | 209 |
| **DOSMerge286** *Specify 2-Users or Unlimited* | 149 | 129 |
| **DOSMerge386** *2-Users* | 395 | 345 |
| **DOSMerge386** *Unlimited Users* | 495 | 429 |
| **System V/386 Combination** | 799 | 669 |
| **386 Runtime System** | 199 | 169 |
| **386 Software Development System** | 499 | 429 |
| **Text Preparation System** | 199 | 169 |
| **System V/AT Combination** | 549 | 465 |
| **AT Runtime System** | 199 | 169 |
| **AT Software Development System** | 249 | 209 |
| **Text Preparation System** | 199 | 169 |

## microsoft products

| | | |
|---|---|---|
| **Microsoft BASIC Compiler** *for XENIX* | 695 | CALL |
| **Microsoft BASIC Interpreter** *for XENIX* | 350 | CALL |
| **Microsoft C Compiler 5.0** *w/CodeView* | 450 | CALL |
| **Microsoft COBOL Compiler** *with COBOL Tools* | 700 | CALL |
| *for XENIX* | 995 | CALL |
| **Microsoft Excel** | 495 | CALL |
| **Microsoft FORTRAN** *Optimizing Compiler* | 450 | CALL |
| **Microsoft FORTRAN** *for XENIX* | 695 | CALL |
| **Microsoft Learning DOS** | 50 | CALL |
| **Microsoft MACH 20** *New* | CALL | CALL |
| **Microsoft Macro Assembler** | 150 | CALL |
| **Microsoft Mouse** *Specify Serial or Bus* | | CALL |
| *with Paint & Mouse Menus* | 150 | CALL |
| *with Microsoft Windows & Paint* | 200 | CALL |
| *with EasyCAD* | 175 | CALL |
| **Microsoft Pascal Compiler** | 300 | CALL |
| *for XENIX* | 695 | CALL |
| **Microsoft QuickBASIC** | 99 | CALL |
| **Microsoft QuickC** | 99 | CALL |
| **Microsoft Windows** | 99 | CALL |
| **Microsoft Windows 386** *New* | 195 | CALL |
| **Microsoft Windows Development Kit** | 500 | CALL |
| **Microsoft Word** | 450 | CALL |
| **Microsoft Works** | 195 | CALL |

## mks products

| | | |
|---|---|---|
| **MKS AWK** | 75 | 65 |
| **MKS RCS** *Revision Control System* | 189 | 155 |
| **MKS Toolkit** *with MKS VI Editor* | 139 | 109 |
| **MKS Trilogy** *with AWK, CRYPT & Korn Shell* | 119 | 99 |
| **MKS VI** *Editor by MKS* | 75 | 65 |

## modula-2 language

| | | |
|---|---|---|
| **LOGITECH Modula-2 Development System** | 249 | 199 |
| **Modula-2 Compiler Pack** | 99 | 79 |
| **Modula-2 Toolkit** | 169 | 139 |
| **LOGITECH Modula-2 Window Pkg** | 49 | 39 |
| **Macro2** *Macro preprocessor by PMI* | 89 | 79 |
| **ModBase** *by PMI* | 89 | 79 |
| **ModGraph** *by TEQNA* | 50 | 45 |
| **MODULA-2** *by Stony Brook* *New* | 195 | 169 |
| *with Utilities* *New* | 345 | 299 |
| **Repertoire** *by PMI* | 89 | 74 |
| **Science & Engrg Tools** *by Quinn-Curtis* | 75 | 67 |
| **Universal Graphics Library** *by Quinn-Curtis* | 130 | 119 |

## mouse products

| | | |
|---|---|---|
| **LOGIMOUSE BUS** *with PLUS Pkg by LOGITECH* | 119 | 98 |
| *with PLUS & PC Paintbrush* | 149 | 119 |
| *with PLUS & CAD Software* | 189 | 153 |
| *with PLUS & CAD & Paint* | 219 | 179 |
| *with PLUS & First Publisher* | CALL | CALL |
| **LOGIMOUSE C7** *with PLUS Package* | 119 | 98 |
| *with PLUS & PC Paintbrush* | 149 | 119 |
| *with PLUS & CAD Software* | 189 | 153 |
| *with PLUS & CAD & Paint* | 219 | 179 |
| *with PLUS & First Publisher* | CALL | CALL |
| **Microsoft Mouse** *See Microsoft Section* | | |

## novell products

| | | |
|---|---|---|
| **Btrieve** *ISAM Mgr with No Royalties* | 245 | 184 |
| **Xtrieve** *Query Utility* | 245 | 184 |
| **Report Option** *for Xtrieve* | 145 | 99 |
| **Btrieve/N** *for Networks* | 595 | 454 |
| **Xtrieve/N** | 595 | 454 |
| **Report Option/N** *for Xtrieve/N* | 345 | 269 |
| **XQL** | 795 | CALL |

## other languages

| | | |
|---|---|---|
| **ACTOR** *by Whitewater Group* | 495 | 419 |
| **CCS MUMPS** *All varieties by MGlobal* | CALL | CALL |
| **Marshal Pascal** *by Marshal Language Systems* | 189 | 148 |
| **Pascal-2** *by Oregon Software* | 395 | 289 |
| **Personal REXX** *by Mansfield Software* | 125 | 99 |
| **SNOBOL4+** *by Catspaw* | 95 | 80 |

## other products

| | | |
|---|---|---|
| **Carbon Copy Plus** *by Meridian Technology* | 195 | 135 |
| **Dan Bricklin's Demo Pgm** *by Software Garden* | 75 | 57 |
| **Dan Bricklin's Demo Tutorial** | 50 | 45 |
| **Fast Forward** *by Mark Williams* | 70 | 59 |
| **HI-Screen XL** *by Softway* *New* | 149 | 129 |
| **Instant Replay III** *by Nostradamus* | 150 | CALL |
| **MicroTEX** *Typesetting from Addison-Wesley* | 295 | CALL |
| **Printer Drivers** | CALL | CALL |
| **muMATH** *by Soft Warehouse* | 300 | 199 |
| **Net-Tools** *by BCSoft* | 149 | 129 |
| **Norton Guides** *Specify Language* | 100 | 65 |
| **OPT-Tech Sort** *by Opt-Tech Data Proc* | 149 | 99 |
| **Peabody** *by Copia Intl, Specify Language* *New* | 100 | 89 |
| **PC-MOS/386** *by The Software Link* *New* | 195 | 155 |
| **PC/TOOLS Deluxe** *by Custom Software..New Version* | 79 | 69 |
| **Resident Expert** *Specify lang by Santa Rita* | 59 | 69 |
| **Screen Machine** *by MicroHelp* | 79 | 59 |
| **screenplay** *by Flexus, Specify Compiler* | CALL | CALL |
| **SuperSort** *by LifeStyle* | 139 | 119 |
| **Teamwork/PCSA** *by Cadre Technologies* *New* | 995 | 929 |
| **The SLATE System** *by The Symmetry Group* *New* | 299 | CALL |

## phoenix products

| | | |
|---|---|---|
| **C/Pac** *Combination of PforCe and Pre-C* *New* | 495 | 279 |
| **Pasm86** *Macro Assembler version 2.0* | 195 | 108 |
| **Pdisk** *Hard Disk & Backup Utility* | 145 | 99 |
| **Pfantasy Pac** *Phoenix Combo* | 995 | 595 |
| **Pfinish** *Execution Profiler* | 395 | 209 |
| **Pfix86plus** *Symbolic Debugger* | 395 | 209 |
| **PforCe** *Specify C Compiler* | 395 | 209 |
| **PforCe++** *Specify C Compiler and C++* | 395 | 209 |
| **Plink86plus** *Overlay Linker* | 495 | 275 |
| **Pmaker** *Make Utility* | 125 | 78 |
| **Pmate** *Macro Text Editor* | 195 | 108 |
| **Pre-C** *Lint Utility* | 295 | 154 |

## polytron products

| | | |
|---|---|---|
| **PolyBoost II** *Software Accelerator* *Special* | 80 | 54 |
| **PolyDesk III** | 99 | 72 |
| **PolyDesk Archivist** | 50 | 42 |
| **PolyDesk Cryptographer** | 50 | 42 |
| **PolyDesk Talk** | 50 | 42 |
| **PolyLibrarian** *Library Manager* | 99 | 89 |
| **PolyLibrarian II** *Library Manager* | 149 | 129 |
| **PolyMake** *UNIX-like Make Facility* | 149 | 129 |
| **PolyShell** *Special Price* | 149 | 105 |
| **Polytron C Beautifier** | 50 | 45 |
| **PolyXREF** *Complete Cross Ref Utility* | 219 | 185 |
| **PolyXREF** *One language only* | 129 | 109 |
| **PVCS Corporate** *Version Control System* | 395 | 329 |
| **PVCS Personal** | 149 | 129 |

## program mgmt utilities

| | | |
|---|---|---|
| **Interactive EASYFLOW** *by Haventree* | 150 | 125 |
| **PrintQ** *by Software Directions* | 89 | 84 |
| **Quilt Computing Combo** *QMake & SRMS* | 250 | 199 |
| **Sapiens MAKE** | 179 | 155 |
| **Sapiens MAKE & V8** | 439 | 379 |
| **Source Print** *by Aldebaran Labs* | 97 | 74 |
| **TLIB** *Version Control System by Burton* | 100 | 89 |
| **Tree Diagrammer** *by Aldebaran Labs* | 77 | 59 |

## sco products

| | | |
|---|---|---|
| **Complete XENIX System V** *by SCO* | 1295 | 979 |
| **Development System** | 595 | 479 |
| **Operating System** *Specify XT or AT* | 595 | 479 |
| **Text Processing Package** | 195 | 144 |
| **Lyrix** *by SCO* | 595 | 449 |
| **SCO Professional** *1-2-3 Workalike for XENIX* | 795 | 595 |
| **SCO XENIX-NET** | 595 | 495 |
| **XENIX System V 386** *by SCO* | CALL | CALL |

## text editors

| | | |
|---|---|---|
| **Brief & dBrief Combo** *from Solution Systems* | 275 | CALL |
| **Brief** | 195 | CALL |
| **dBrief** *Customizes Brief for dBASE III* | 95 | CALL |
| **Epsilon** *Emacs-like editor by Lugaru* | 195 | 147 |
| **KEDIT** *by Mansfield Software* | 125 | 98 |
| **Micro/SPF** *by PHASER SYSTEMS* | 175 | 139 |
| **Microsoft Word** | 450 | CALL |
| **PC/VI** *by Custom Software Systems* | 149 | 99 |
| **SPF/PC** *by Command Technology Corp* | CALL | CALL |
| **Vedit Plus** *by CompuView* | 185 | 128 |

## turbo pascal utilities

| | | |
|---|---|---|
| **ALICE** *Interpreter by Looking Glass Software* | 95 | 66 |
| **AZATAR DOS Toolkit** *by AZATAR* | 95 | 85 |
| **DOS/BIOS & Mouse Tools** *by Quinn-Curtis* | 75 | 67 |
| **Flash-up** *by Software Bottling* | 89 | 78 |
| **Flash-up Developer's Toolbox** | 49 | 45 |
| **MACH 2** *for Turbo Pascal by MicroHelp* | 69 | 55 |
| **MetraByte D/A Tools** *by Quinn-Curtis* | 100 | 89 |
| **Science & Engrg Tools** *by Quinn-Curtis* | 75 | 67 |
| **Screen Sculptor** *by Software Bottling* | 125 | 91 |
| **Speed Screen** *by Software Bottling* | 35 | 32 |
| **System Builder** *by Royal American* | 150 | 129 |
| **IMPEX** *Query Utility* | 100 | 89 |
| **Report Builder** | 130 | 115 |
| **TDebugPLUS** *by TurboPower Software* | 60 | 49 |
| **Tmark** *by Tangent Designs* | 80 | 69 |
| **Turbo Professional 4.0** *from TurboPower... New Vrs* | 99 | 79 |
| **TurboHALO** *from IMSI* | 95 | 75 |
| **TurboPower Utilities** *by TurboPower* | 95 | 78 |
| **TurboRef** *by Gracon Services* | 50 | 35 |
| **TURBOsmith** *Source Debugger by Visual Age* | 99 | 69 |
| **Universal Graphics Library** *by Quinn-Curtis* | 130 | 119 |

## wendin products

| | | |
|---|---|---|
| **Operating System Toolbox** | 99 | 75 |
| **PCNX** *Operating system* | 99 | 75 |
| **PCVMS** *Similar to VAX/VMS* | 99 | 75 |
| **Wendin-DOS** *Multitasking DOS* | 99 | 79 |
| **Wendin-DOS Application Developer's Kit** | 99 | 79 |
| **XTC** *Text Editor w/Pascal source* | 99 | 75 |

## xenix/unix products

| | | |
|---|---|---|
| **Btrieve** *ISAM File Mgr by Novell* | 595 | 454 |
| **C-terp** *by Gimpel* | 498 | 379 |
| **c-tree** *ISAM Mgr by FairCom* | 395 | 315 |
| **dBx** *with Library Source by Desktop AI* | 550 | 419 |
| **DIRECTORY SHELL 286** *by American Mgmt Sys* | 349 | 295 |
| **DIRECTORY SHELL 386** *by American Mgmt Sys* | 495 | 415 |
| **Epsilon** *Text Editor by Lugaru* | 195 | 147 |
| **PANEL Plus** *by Roundhill Computer Systems* | 795 | 535 |
| **REAL-TOOLS** *Binary Version by PCT* | 99 | 89 |
| *Complete Source Version* | 999 | 729 |
| **RM/COBOL** *by Ryan-McFarland* | 1250 | 949 |
| **RM/FORTRAN** *by Ryan-McFarland* | 750 | 549 |

Terms are subject to change.



**CIRCLE NO. 198 ON READER SERVICE CARD**

### LOWEST PRICES

Due to printing lead times, some of our current prices may differ from those shown here. Call for latest pricing.

### FREE SHIPPING

Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

### CREDIT CARDS

VISA and MasterCard are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and authorized signature.

### CODs AND POs

CODs and Purchase Orders are accepted at no extra cost. No personal checks are accepted on COD orders. POs with net 30-day terms (with initial minimum order of $100) are available to qualified US accounts only.

### SALES TAX

Orders outside of Ohio are not charged state sales tax. Ohio customers please add 5% Ohio tax or provide proof of tax-exemption.

### INTERNATIONAL ORDERS

Shipping charges for International and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. International orders (except Canada), please include an additional $10 for export preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

### VOLUME ORDERS

Volume orders may qualify for additional discounts. Call us for special pricing.

### SOUND ADVICE

Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

### 30-DAY GUARANTEE

Most of our products (excluding books) come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

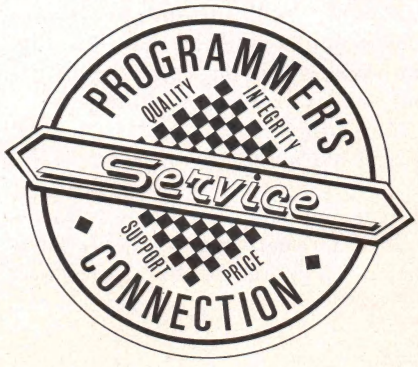
### MAIL ORDERS

Please include your telephone number on all mail orders. Be sure to specify computer, operating system and any applicable compiler or hardware interface(s). Send mail orders to:

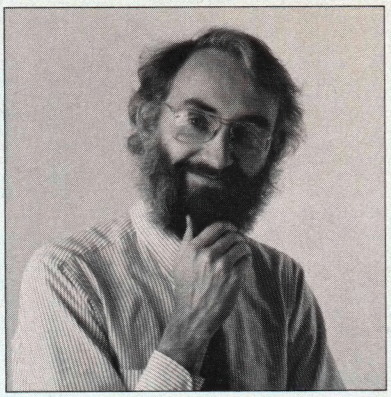
**Programmer's Connection**
**7249 Whipple Ave. NW**
**North Canton, OH 44720**

| | |
|---|---|
| **USA** | **800-336-1166** |
| **CANADA** | **800-225-1166** |
| **OHIO & ALASKA (Collect)** | **216-494-3781** |
| **TELEX** | **9102406879** |
| **EASYLINK** | **62806530** |
| **INTERNATIONAL** | **216-494-3781** |
| **CUSTOMER SERVICE** | **216-494-8899** |

**Hours: Weekdays 8:30 AM to 8:00 PM EST.**



# SWAINE'S FLAMES



True magazines really do have something like personality. You can develop a relationship with them. That's why it seems so unfair when a publisher repositions an old friend, as McGraw-Hill is doing with *Byte*. It's all the more annoying when the change seems so patently unnecessary. Isn't it obvious that we need a Scientific American of computer technology more than we need another PC magazine? Not to McGraw-Hill it isn't.

McGraw-Hill is not alone in opting for the product-catalog approach; the trend is depressingly widespread. Even our new section of short reviews, Examining Room (debuting in this issue), could be seen as symptomatic of the trend. But *DDJ* is, I assure you, in no danger of becoming a product catalog. It wouldn't fit with the personality.

Speaking of personality, the Peter Norton T-shirt came in the mail recently, close on the heels of the Peter Norton poster, the Peter Norton Christmas card, and the Peter Norton coffee cup. (If you didn't like the coffee cup, Peter Norton would send you a Peter Norton hammer to smash it with.)

Now I believe that the Norton Utilities are nifty, and that Peter has as much right as anybody to throw his hat in the ring for Mr. Cultural Icon of 1988, but all this fuss over a company that produces programs for wiping files and cleaning up directories gives me pause. Norton is, after all, in the utility business, a sort of electronic Tidy Bowl man, the digital descendant of Ed Norton. Is this an image you want to display on your coffee table? Consider well when marking your ballot.

Another clean-cut kid, Rob Dickerson, has been making news lately, in the sections of what newspaper business pages call Transitions. Some of you know Dickerson from Microsoft's language division. In December he became Borland's latest acquisition, with the new title of vice president, product marketing. Microsoft responded to Dickerson's career move with a lawsuit and a restraining order, and Borland countersued. By the time you read this the suits should be settled, or at least moot.

Microsoft had reason to be upset; Dickerson knows a lot that could be useful to the competition, and that means Borland. (I am specifically not referring to his knowledge of Microsoft strategies. I have no knowledge regarding Rob's knowledge about Microsoft strategies, thank you, Microsoft lawyers. What I have is informed opinion.) Meanwhile, Borland is building new quarters in Scotts Valley to accommodate its growth. I think it is going to call the new space the Borland campus.

I spent many pages in this magazine last year campaigning for articles on how to break various bandwidth bottlenecks. Bandwidth is, unfortunately, becoming a vogue word. You can find it in John Sculley's *Odyssey: From Pepsi to Apple*, where Sculley applies it to marketing. He probably picked up this usage from Steve Jobs, a good man with a metaphor.

You can find the term used more precisely and with more purpose in *The Media Lab* by Stewart Brand, a good book on current research at MIT into the convergent technologies of publishing, computing, and broadcasting. The principals in *The Media Lab* believe that bandwidth reduction is the issue of the decade, and they are acting on that belief. I recommend Brand's book. (I haven't been inspired to read Sculley's yet.)

I sometimes think that Marlin Ouverson, a former *DDJ* editor, timed his leaving so that he could title his farewell editorial, in paraphrase of Richard Henry Dana, "Two Years Before the Masthead."

Which leads me to the following.

I became editor-in-chief of *Dr. Dobb's Journal* in 1984 and have now spent more years at the helm than any other editor (including Marlin Ouverson). While I am fond of this magazine and think it serves a need that the product-catalog magazines don't, four years is a *long* time—a third of the life of the magazine, a tenth of my own life, longer than Sculley has been at Apple (or about as long as there has been a Macintosh product line)—and these four years have been four years away from full-time writing on the rock pile of a largely administrative job.

For a writer, that's a long sentence.

So, over the next few months, I will be creating a new role for myself with the magazine, one that will allow me to do more writing. I plan to research and write about topics close to my interests and, according to our reader survey and to my informal survey in the November issue, close to yours as well.

More about that next month.

Michael Swaine

Michael Swaine
editor-in-chief

# Some Of The Most Famous Faces In Software Use Our C Functions

Our famous customers are a little camera shy. It's not that they are embarrassed by being Essential C Utility Library users. They just don't want us shouting their names from the roof tops.

The prestige of our users is not the primary reason to buy the Essential C Utility Library. The increased speed, features, and size efficiency of our products are the factors that demand their use. Our library contains *over 400 functions,* all designed with elegance in mind.

However, should curiosity get the best of you, call us at 201-762-6965 and we'll drop a few highly impressive names on you.


*Behind every great program is a great library.*

## Functions At A Glance

- Fastest screen output available.
- Save/Restore color screens in 1/10 sec.
- Pop-up block cursor menus
- Save/Restore windows to disk or memory
- 50 functions for business graphics
- dozens of string formats
- time and date arithmetic
- julian and day-of-week
- Ctrl-Break key trapping
- Field oriented data entry
- Stuff keyboard buffer
- 18 Mouse control functions
- Execute programs and batch files
- Disk error trapping
- Determine space available
- 40 functions to process characters and words
- Insert, delete, extract, index, translate
- Tested, easy-to-follow examples
- Demo programs with source code
- All source code included

**Documentation:**
Thorough, comprehensive, 260 pages

**Compatible C Compilers:**
Microsoft, Lattice, Computer Innovations, Aztec, Mark Williams, DeSmet, and Wizard

**$185.00**

## Do Your Homework

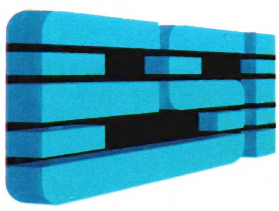
The library you buy can influence the rest of your programming life. We encourage you to do some checking before making a decision. When you've done your homework, you'll choose Essential. Call our support staff of experienced C programmers and find out before you buy how things will be after your check clears.

## What's a Library Without a Librarian?

Our library comes complete with a sophisticated source code librarian. Now you can maintain current versions and conserve disk space. We want your development work to go as smoothly as possible.

## No Royalties, 30 Day Guarantee

If within 30 days you don't find our library totally satisfactory, bag the whole thing and receive a complete refund. There are no royalties associated with the library.



**To order or for support call: 201-762-6965**

**For foreign orders contact:**
England: Gray Matter Tel. (0364) 53499
Japan: Lifeboat Inc. of Japan Tel: 293 4711
West Germany: Omnitex Tel. 07623-61820

## Essential Software, Inc.

P.O. Box 1003, Maplewood, New Jersey 07040

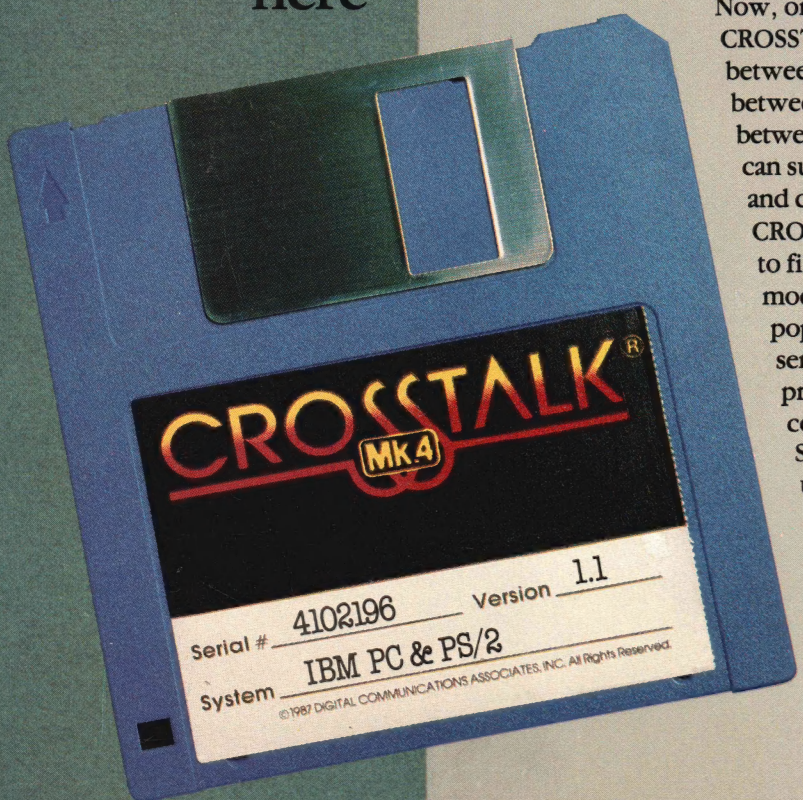
CIRCLE NO. 199 ON READER SERVICE CARD

# IBM Spoken Here

and here

and here

and here




**Whatever dialect of IBM you need to speak, CROSSTALK® Mk. 4 makes the connection.**

Now, one program does the job that used to require several. CROSSTALK® Mk. 4 allows high-speed direct communications between PCs and minicomputers, or (with an IRMA™ board) between your PC and an IBM Mainframe, or (with Smart Alec™) between your PC and IBM System 3x's. If you like, CROSSTALK can support all of these sessions (and others) simultaneously, and display each session in its own window.

CROSSTALK Mk. 4 emulates all the terminals you're likely to find useful. That includes IBM 3101 (page and character modes), IBM 525x, IBM 529x, IBM 327x, as well as many popular async terminals like the DEC VT100 and VT220 series. CROSSTALK Mk. 4 includes the powerful CASL™ programming language, which allows you to automate communications applications quickly and easily.

So if you're used to thinking of CROSSTALK just to use with a modem, you're missing some important connections. Ask your dealer for details, or write:

dca® Digital Communications Associates, Inc.
1000 Holcomb Woods Parkway / Roswell, Georgia 30076
1-800-241-6393

CROSSTALK® COMMUNICATIONS



CIRCLE NO. 200 ON READER SERVICE CARD